The Geologic Retrieval and Synopsis Program (GRASP)

By ROGER W. BOWEN *and* JOSEPH MOSES BOTBOL

GEOLOGICAL SURVEY PROFESSIONAL PAPER 966

A portable data-retrieval system requiring minimal user training

UNITED STATES GOVERNMENT PRINTING OFFICE, WASHINGTON : 1975

UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

V. E. McKelvey, *Director*

Library of Congress Cataloging in Publication Data

Bowen, Roger W.
The geologic retrieval and synopsis program (GRASP)

(Geological Survey professional paper; 966)
Supt. of Docs. no.: I 19.16:966
1. Information storage and retrieval systems--Geology. I. Botbol, Joseph Moses, joint author. II. Title. III. Series: United States. Geological Survey. Professional paper; 966.
QE48.8.B68 029'.9'55 75-619314

For sale by the Superintendent of Documents, U.S. Government Printing Office
Washington, D.C. 20402
Stock Number 024-001-027-19-4

CONTENTS

ILLUSTRATIONS

THE GEOLOGIC RETRIEVAL AND SYNOPSIS PROGRAM (GRASP)

By Roger W. Bowen and Joseph Moses Botbol

ABSTRACT

The Geologic Retrieval and Synopsis Program (GRASP) was designed and written to specifically accommodate interactive access to earth-science data banks. GRASP is portable, easy to use, and data-base independent.

Data banks accessed by GRASP must be partitioned and reformatted into five files which make up both data bank and pointers to parts of the bank. Machine dependencies include FORTRAN I/O unit numbers, direct-(random) access input, in-core read/write, and "prompting." GRASP isolates these dependencies to FORTRAN subroutines designed to serve these functions specifically. GRASP is manipulated by 11 user commands which select, describe, access, retrieve, summarize, and display data.

INTRODUCTION

The U.S. Geological Survey presently has the responsibility of developing and maintaining resource-data banks. Initially, many storage and retrieval systems were critically reviewed for their data-bank characteristics, ease of use, flexibility, portability, and applicability to Survey activity. The authors concluded that no available system was wholly adequate for the needs of the Survey data banks. Generally, the observed systems were difficult to use, machine bound, or were oriented toward one type of data (for example, text oriented). The only logical alternative was to design, develop, and implement a geologic-data storage and retrieval system to be used primarily by geologists.

PURPOSE AND SCOPE

The Geologic Retrieval and Synopsis Program (GRASP) was written to provide a means of interactive access to geologic data stored in a time-sharing computer.

GRASP can be implemented on any time-sharing computer system that has a FORTRAN IV compiler. Data bases accessed by GRASP must contain fixed field data in alphameric, alphanumeric, and/or numeric modes.

GENERAL SYSTEM DESCRIPTION

To obtain a broad overview of the GRASP system, consideration should first be given to the philosophy governing the design of the system, use of the present system, and future system plans. This overview provides the "framework" and the anticipated "operational environment" which are necessary for the development of any system. Both the present utilization and future system plans show the correctness of the parameters and techniques used as well as the original assumptions regarding "how," "by whom," and "where" the system will be used.

DESIGN PHILOSOPHY

Three vital questions that must be answered prior to the implementation of any system are: (1) On what computer(s) will this system be used? (2) What are the characteristics of the data to be processed? (3) Who will use the system? As is the case in many computer-system applications, these questions originally had no definite answers. A 3-month effort was necessary to establish criteria that would govern GRASP design.

MACHINE PORTABILITY

To serve as broad a spectrum of the scientific community as possible, a system should be as portable as possible. Because of differences in computer design, no system can be used on all computers without some modification. ANSI FORTRAN IV is universally accepted as a standard programming language. It may be used on the vast majority of present-day computers that have the capacity for implementing compiler-level languages in a time-sharing mode of operation. For this reason, all the processing subroutines in GRASP are written in ANSI FORTRAN IV (machine-dependent features isolated to facilitate implementation). In this way, GRASP can be installed on virtually any modern

time-sharing computer. By designing GRASP to be portable, a much wider spectrum of the scientific community can be served by the system. Data need not be transferred from their resident banks in order to be accessed. GRASP could be used on most computers in order to access data files wherever they may reside. This machine independence eliminates the need for tedious data transformations to one system configuration, where the aggregating data eventually flood to the point of uselessness the peripheral storage of a central machine-dependent data bank. The authors believe that a common accessing system for data residing in different computers is preferable to an accessing system that can be used for data resident in only one computer.

DATA-BASE INDEPENDENCE

GRASP is designed to operate using any data base that can be represented in conventional matrix form. In matrix form, the records (that is, items to be described) are the rows, and the attributes of each record are the columns. The real structure of the data base can be thought of as the titles and arrangement of the columns of a data matrix where the rows are merely instances or occurrences described by the columns. For example, a geochemical data base would have a matrix representation in which the columns might represent chemical analyses of various elements, and each row would represent one sample. GRASP can function on any data matrix. The only requirement is that the variables (or columns) be defined ahead of time in terms of their types (that is, alpha or numeric), and, where necessary, dictionaries of legitimate alphameric entries must be provided for alphameric variables. Thus, because of the matrix orientation of GRASP, any fixed field data base can be accommodated by the system.

USER COMMUNITY

GRASP is a retrieval system having its own rules and command language for operation. In other words, to use GRASP, the user does not need to be familiar with FORTRAN or any other computer language. The GRASP command language is designed to provide users with the ability to ask questions of a data base and retain all items that answer "true" to the questions. The control language used to ask the questions (discussed in the section on "Use of GRASP" in this report) allows "retrieve only" data access to any GRASP user and does not require prior user knowledge of computer languages or system functions. Thus, GRASP can be implemented for a wide variety of users.

TIME SHARING

Inasmuch as GRASP is portable, data-base independent, and serves a wide variety of users, GRASP should be implemented in a computational mode that has the most readily available user access, namely, time sharing. In its simplest form, time sharing allows any user to communicate with the computer from a terminal near a telephone. The entire design of GRASP is based on the premise that the user community will converse directly with a computer (via a terminal) in order to access, retrieve, manipulate, summarize, and display data. This mode of computation provides the "instant" response necessary for timely decisionmaking, and also allows access by the user from the environment in which the computer response is of most value, that is, the laboratory, field, office, or conference room.

PRESENT UTILIZATION

In 1975, the GRASP system was being used to access data from six totally different data banks: (1) oil- and gas-pool characteristics of Colorado, (2) mineral deposits of the world, (3) geochemical exploration data from the United States, (4) coal resources of the United States (prototypic data bank), (5) index of U.S. geologic map coverage (prototypic system), and (6) geothermal data bank (in Pisa, Italy). The first three of the above systems were implemented directly by the authors, and no attempt was made to redesign any of the original data-bank structures.

COLFIL

This file contains as many as 390 characteristics for each of 800 oil and (or) gas pools in Colorado. This file served as the original model for GRASP design and ultimately will contain 60,000 records.

MANFIL

The mineral deposits of the world (MANFIL) were the second file implemented using GRASP. It is a computerized batch-processing-oriented file containing geologic, production, and reserves data from about 4,000 nonferrous metal deposits throughout the world. Each record represents one deposit, and contains as many as 250 variables. Although GRASP was designed using the oil- and gas-pool file as a model, implementation of the world-mineral-deposits file showed the flexibility of GRASP with respect to its data-base independence.

RASS

The RASS (Rock Analysis Storage System) file is a batch-oriented geochemical data bank containing limited geologic descriptions and comprehensive geochemical analyses of all samples processed by the laboratories of the U.S. Geological Survey. This file contains a unique type of numeric data called "qualified values." Because of the upper and lower detectability limits of analytic devices, elements whose presence is known but whose content is outside the analytical range of a device are sometimes reported at a given analytical cutoff value, accompanied by a letter indicating whether the content is less than, greater than, or in interference with another element. Typical qualified entries would appear as L5000, G1000, or H100, where L signifies a content less than the attached value, G signifies a content greater than the attached value, and H signifies analytical interference at a concentration of the attached value. Because many of the RASS data were accompanied by alpha qualifiers, GRASP was modified to accept and process this type of data in addition to the conventional numeric- and alphameric-data types.

All the above files are implemented in a retrieve mode only, and graphics have not yet been added. Input to the files is done by people who are responsible for data entry and does not fall in the domain of the user.

FUTURE PLANS

Currently, the development of GRASP is primarily oriented toward implementation of techniques for interactive graphics storage and retrieval. Three problem areas are presently being researched: automatic recognition of features on scanned input documents, annotation methods, and resolution of "intersecting feature" problems. Present research efforts are directed toward feature recognition and subsequent computational extraction of simple boundary vectors from scanned digital maps and photographs. In addition to recognition of features, methods are also being developed for annotation of both graphics-data entry and presentation of graphics data on output.

One of the major anticipated technical and philosophical problems is concerned with the graphical resolution of intersecting features. Techniques are being developed that should resolve these problems for any particular data set.

All the GRASP graphics output is being designed primarily for interactive graphics cathode ray tube (CRT) representation. This is in keeping with GRASP's original "totally interactive" design philosophy.

DETAILED SYSTEM DESCRIPTION

GRASP is designed as a highly modular, hierarchically structured set of subroutines (see section "GRASP Software Specifications"). Each subroutine performs a fixed task. The higher level subroutines are primarily concerned with the flow of control required to execute a user command. The lower level subroutines are primarily concerned with extremely independent and specific tasks (such as "get a record," "access a dictionary," and "accept user input," and so on). All information related to a specific data base is obtained from various files associated with that data base. Structuring the system in this way leads to a high degree of functional isolation. These design characteristics simplify the development, documentation, maintenance, growth, and inevitable change inherent in a system that supports a variety of data bases on a wide spectrum of computer main frames. The section on GRASP software specifications is intended for use by those familiar with FORTRAN language.

DATA-FILE STRUCTURES

Upon initial execution, the GRASP system reads an "index" file which contains the names of data bases available for access. Each record of the index file corresponds to a data base and contains the names of the files associated with that data base and a 40-character description of the data base.

Each data base is composed of five files which contain the actual data, information on the structure of records, names which will be used to refer to particular items within records, descriptive information on the names themselves, and a grouping into categories of information. These files are called Mask, Definitions, Dictionary, Multiple-choice, and Numeric Master files.

MASK FILE

The Mask file contains the item names, item types (integer, real, character string, multiple choice, and qualified real), and pointers to the first entry in the Dictionary file for each character-type item. This file is read once and rewound when a data base is selected via the FILE command. An example of Mask file arrangement is shown in figure 1.

Conceptual noncomputerized dictionaries for three character type variables:

Variable No. 1	Variable No. 2	Variable No. 3
Continent	Country	Province (State)
North America	USA	California
South America	Canada	Virginia
Europe	Mexico	British Columbia
	Argentina	Quebec
	Brazil	Cordova
	Chile	
	Germany	

In the computer, the MASK File is arranged as follows:

Mask File

Variable Name	Variable Type	Starting Position in Dictionary File
Continent	Character	1
Country	Character	4
Province (State)	Character	11
Production	Numeric, real	*

*Note: This variable is numeric, and does not require a pointer to the Dictionary File

In the computer, the Dictionary File is arranged as follows:

Dictionary File

Item No.	Pointer to next item	Name
1	2	North America
2	3	South America
3	0	Europe
4	5	USA
5	6	Canada
6	7	Mexico
7	8	Argentina
8	9	Brazil
9	10	Chile
10	0	Germany
11	12	California
12	13	Virginia
13	14	British Columbia
14	15	Quebec
15	0	Cordova

("0" indicates end of list for particular variable)

FIGURE 1.—Example of Mask-file and Dictionary-file arrangement.

DEFINITIONS FILE

The Definitions file contains the following information:

1. The number of categories in the file.
2. The maximum number of (computer) words in a category name.
3. The category names.
4. For each category the following information is recorded:
   (a) category number.
   (b) number of lines used to describe this category.
   (c) maximum length (in computer words) of a description in this category.
   (d) number of variables appearing in this category. In some cases this will be different from item b (the number of lines for description).
   (e) indices of the variables appearing in this category.
   (f) the variable names, types, and descriptions for this category.

DICTIONARY FILE

The Dictionary file contains all character-string values which are assumed by character-type items. Each record contains a pointer to the record containing the next value, followed by the current value. The last value assumed by a character-type item is indicated by a pointer value of zero (the record containing the first character-string value for a character-type item is pointed to by a value in the Mask file). The Dictionary file is designed as a random-access file whose values form a linked list. Figure 1 shows an example of the Dictionary-file arrangement.

MULTIPLE-CHOICE FILE

The Multiple-choice file contains the acronyms and acronym meanings for the values assumed by mul-

tiple-choice items. Each record of this file is composed of an item number indicating the multiple-choice item, the number of possible values this item assumes, the maximum length of an acronym value description, and a list of acronyms (which are double words) and their descriptions.

NUMERIC MASTER FILE

The Numeric Master file is composed of the records for a data base in a compressed form. Values for integer-type items are stored as integers. Values for floating-point- (or real-) type items are stored as real numbers. Values for character-type items are stored as integer pointers to the entry number in the Dictionary file. Values for multiple-choice-type items are stored as integers containing a binary encoding that represents the value set. (For example, if the second and fifth bit of the word are "on," the value assumed is the second and fifth acronym value.) Each record of the compressed Master file is variable length in form and corresponds to an expanded 400-word record. Expansion of the compressed record is performed by subroutine GETPUT. Figure 2 shows an example of the Numeric Master file prior to compression.

DATA COMPRESSION

The compression technique used is a form of blank suppression. The words of the compressed record are one of the following four types:

A. Integer value.
B. Real value.
C. Integer blank count.
D. Integer word count.

The first word of all records is of type D (above) and is used to give the length of the record. Subsequent words may be types A, B, and C. For types A and C, the last two bits give the type of the next word. The value of the word is obtained by dividing by 4. The type of the next word is obtained via the remainder modulo 4, where the numbers 1–3 correspond to types A, B, and C. Type-A words are used for numeric integers, pointers to entries in character dictionaries, and binary encodings of multiple-choice-type items. Type-C words are used to count the number of consecutive blanks to be inserted in the expanded record. Type-B words are used for floating-point numeric values. The type of the next word is contained in the last 2 bits of the whole (integer) part of the words. For example, consider a data word having a value of 49.723. The

Given the following two successive noncomputerized records to be entered into the Numeric Storage File:

	Record 1	Record 2
Continent	North America	South America
Country	United States	Argentina
Production	39281.6	49298.7
Identification No.	38	39
Province (or State)	Virginia	Cordova

Prior to compression, the computerized Numeric Storage File is arranged as follows:

Continent*	Country*	Production	ID No.	Province*	etc.
1	1	39281.6	38	2	
2	4	49298.7	39	5	

*See figure 1—for dictionary codification of continent, country, and province

FIGURE 2.—Example of Numeric-Master-file arrangement prior to compression.

floating-point value would be 12.723 (12=49/4), and the type of the next word would be 49−(4×12)=1 or A.

MACHINE DEPENDENCIES

Although most of the GRASP-system code is written in ANSI FORTRAN IV, certain isolated functions must be tailored to the particular FORTRAN compiler on any given machine. These functions deal with the dynamic association of data set names and FORTRAN I/O unit numbers, direct- (random-) access input, the method of accommodating "prompting," and internal (in-core) transfer (writes) using format control.

For the dynamic association of data-set names and FORTRAN unit numbers, the routines IFILE, OFILE and DEFINE are used. Details of these routines can be found in the section "GRASP Software Specifications."

Direct-access input is used to access the Dictionary file (in subroutine ACCESS). The FORTRAN unit number, an integer expression giving the record number, and an input list are supplied in the READ statement. This form of direct-access input is compatible with most FORTRAN compilers which support direct-access input. Systems not having direct-access capabilities can be accommodated by modifying the logic of this subroutine. This modification involves positioning of a sequential file to the appropriate record prior to execution of the READ statement.

For systems which do not accept the "prompt" option in the READ statement, user "prompting" can be accomplished by using WRITE statements immediately preceding (in time) user input. The "prompt" message is contained in a FORMAT statement, along with a character which inhibits the generation of the normal carriage-return/line-feed usually associated with output directed to a time-share terminal. If a particular system does not have this capability, the "prompting" message will appear on a separate line immediately preceding the user input.

The internal transfer of data under format control is accomplished via the ENCODE statement. The ENCODE statement is used in subroutine COLPNT to construct a line of output. The only other use of ENCODE is in subroutine PACK which is used to convert characters from unpacked to packed form. Most non-IBM FORTRANS support this statement in one form or another. In the case of IBM FORTRAN, a routine must be provided that allows internal data transfer under format control.

INTERNAL STRUCTURE AND FUNCTIONS

GRASP is designed to accept a "command" (or directive) from the user. Once the command has been recognized, the appropriate subroutine is executed. This subroutine will, in most cases, call other subroutines in order to accomplish its intended task. In some cases subroutine calls are nested to a depth of six. Figure 3 gives a pictorial summary of the calling hierarchy for subroutines which are in GRASP. This figure will be useful in implementing or modifying the GRASP system.

PROCESSING USER INPUT

All user input to GRASP is passed to the system in unpacked character form. At the highest level are single words used to execute a GRASP "command." In this case, the characters are packed, and the result is compared to the list of available commands. After a command has been issued, supplementary user input is usually required. This supplementary input must then be "parsed" (that is, converted) into a form more meaningful to the GRASP system. This parsed form is entirely numeric in nature. The numbers themselves may represent values, integer encodings, or pointers. Supplementary input falls into five independent areas: conditional expressions, logical expressions, number lists, name lists, and arithmetic expressions.

CONDITIONAL EXPRESSIONS

A conditional expression is an attribute name, followed by a relation, followed by a value. The attribute name is identified using the binary-search technique. The relation is identified by a sequential table lookup. The value is converted to correspond in type with the attribute name referenced. This may result in a pointer to a character entry in the Dictionary file, a binary encoding of an acronym value in a record of the Multiple-choice file, or simply a numeric value. Each conditional expression entered is associated with a letter (A–Z).

LOGICAL EXPRESSIONS

Logical expressions are composed of letters referring to conditional expressions, the grouping symbols used to control the order of evaluation, and the logical operators .AND. (*), .OR. (+), .NOT. (−). For ease of evaluation logical expressions are converted to reverse-Polish form. This is a parenthesis-free form which permits rapid evaluation using a push-down stack technique. For a detailed description of the conversion to and evaluation of reverse-

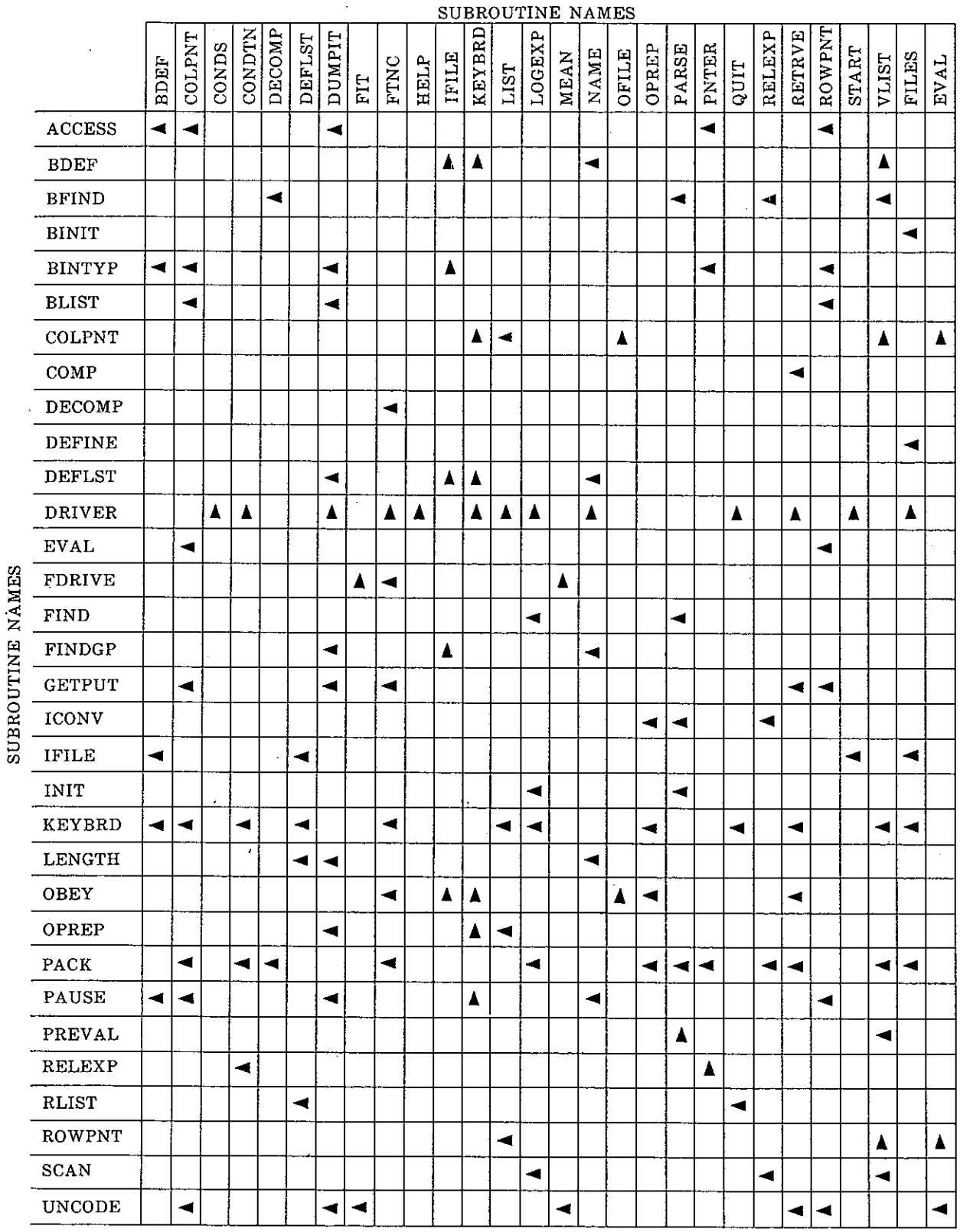

FIGURE 3.—Pictorial summary of GRASP subroutine interrelationships. The arrows point to subroutines that are called.

Polish form expressions, the reader is referred to Lee (1967, p. 162–180).

NUMBER LISTS

Number lists are composed of one or more integer numbers or number ranges (for example, 1, 2–7, 10). Each pair of elements in a number list must be separated by a comma. Individual elements are generated from the unpacked character form by constructing each number, one digit at a time, using a sequential lookup on each character. Number ranges are generated by filling in the interior numbers from the pair of extremes.

NAME LISTS

Name lists are composed of one or more names. When the name list contains a single element such as a file name, the packed form is obtained and associated with the appropriate FORTRAN unit number. If the name list is one or more attribute names, each pair of which is separated by a comma, each element is packed and looked up using the binary-search technique.

ARITHMETIC EXPRESSIONS

Arithmetic expressions may be entered in the place of single attribute names as supplementary input to the LIST command. These arithmetic expressions may be composed of constants, the grouping symbols ( ), attribute names, the arithmetic operators +, *, /, −, and the functions square (SQR), square root (SQRT), log base 10 (LOG) and power of 10 (EXP). The arithmetic expression is converted on input to reverse-Polish form. Evaluation is done on output. If any of the attributes in the expression has no value for a given record, the expression is not evaluated. All conversion to reverse-Polish form is done using transition-matrix parsing. Bauer and Samelson (1960) give a discussion of this technique.

SEARCHES

The two general types of GRASP searches are external file and internal table. External-file searches are made on the Dictionary file, Multiple-choice file, and Numeric Master file.

The Dictionary file is searched in two ways. The first way is as an indexed sequential file. When a condition relating an attribute name to a character-string value is entered, the record number of the first entry of the dictionary for that attribute is obtained from the "unnamed" common area. That record is read (directly), giving the entry value and a pointer to the record containing the next entry. The entry value is compared with the character-string value, and, if not equal, the record containing the next entry is read. This process continues until an entry is found that matches the character-string value, or until all entries of the dictionary have been read. The latter condition is detected by a next-record-pointer of zero. The second way of searching the Dictionary file is as a direct- (random-) access file. To display the value of a character-type attribute, its pointer is obtained from the current record of the selected Numeric Master file [1] and its value is obtained by a direct-access read on the Dictionary file.

Multiple-choice acronym values are obtained when a condition is entered involving a multiple-choice-type attribute and when the value(s) of a multiple-choice-type attribute is to be displayed. Each record of the Multiple-choice file contains all the acronym values for a particular multiple-choice attribute. In all cases, the Multiple-choice file is read in direct- (random-) access mode using a pointer from the "unnamed" common area. After the correct record has been obtained, the attribute values are available in a tabular (array) form.

The Numeric Master file is searched in a purely sequential fashion. This search involves the application of a "question" to each record of the file where the "answer" can only be "yes" or "no." If the answer is "yes," the record is written on an output file. The question is posed by previously entering conditions and relating them by a logical expression.

Internal table searches are made on attribute names and single characters. All lookups on attribute names are done using the "binary search" technique on a sorted list. The list of names are read and sorted by execution of the FILE command. The interval that possibly contains the desired name is repeatedly halved until it is of length one. At that point, the position of the name is known, or the name is not in the list.

Single characters are looked up sequentially when the list of possibilities is short, as in the case of digits in a number. A "hash code" technique is applied for longer lists such as alphabetic letters. This technique involves the initial storage of possibilities in a position dictated by a function applied to the value itself. This is done in a table whose size is greater than the number of possibilities. If

[1] The selected Numeric Master file most probably will be some retrieved subset of the true Numeric Master file.

the position is already occupied, an additional function is applied to the value until an unoccupied position is found. Once the possibilities have been stored in this manner, the lookup of an arbitrary character is accomplished by applying the same procedure. If an empty position is detected during lookup, the character is not in the list. Bell and Kaman (1970) give a more detailed description of the technique.

OUTPUT

A data-retrieval system designed for interactive use should provide the user with information regarding use of the system, the structure and content of a particular data base, and the capability of displaying selective attribute values for records of some partition of a data base. These capabilities have been incorporated in GRASP and are individually discussed in the following paragraphs.

Information regarding use of the GRASP system is provided in two ways. First, all user response is preceded by system-generated "prompts" which indicate the type of response desired. Secondly, a command (HELP) has been implemented that gives the user a brief description of each command recognized by the system.

Information regarding the structure and content of a particular data base is obtainable via the NAMES and FUNCTION commands. The NAMES command allows the user to determine attribute names (acronyms) and corresponding data types. A brief description is provided for each attribute name printed. After the selected attribute names have been printed, the user may examine the set of possible values assumed by character-and multiple-choice-type attributes. For numeric-type attributes, the user may obtain arithmetic means and ranges by selecting the MEAN function after issuing the FUNCTION command.

A partition of a data base is created when a retrieval has been made using the CONDITIONS, LOGIC, and SEARCH commands. The displaying of selective attribute values for records of this partition is accomplished by using the LIST or DUMP commands. The DUMP command permits the user to print all values present for attributes in a selected set of categories. The values are printed one to a line with the corresponding attribute name. The LIST command permits the selection of specific attributes or arithmetic expressions containing attribute names for printing. The printing is selectably formatted into columns or rows. For columnar output, the user may create a separate data set which could be used by other programs at a later time.

DATA-BASE IMPLEMENTATION

In the previous section on file description, it was noted that the various files were integral to and necessary for GRASP to function. There are approximately as many methods of data collection as there are data bases, and it is not the intention of the writers to dictate data-base structures or methods of data collection. However, the following suggestions will facilitate the construction of the files necessary for GRASP implementation.

The structure of any GRASP data base must be such that it can be manifest in a tabular fashion. The table representing a data base is composed of columns that are attributes and rows that are items described by these attributes. For purposes of this report, the word "record" will be used in reference to rows. Before any files can be constructed, a comprehensive list of names of attributes (or column headings) must be compiled. Keep in mind that this arrangement of attributes will describe every record in the data base, and that although provision is made for all attributes, no record need contain data on every attribute. For each attribute that assumes a character-string value, a dictionary is compiled whose entries are the values assumed by that attribute. These dictionaries are used to create the Dictionary file. Once the Dictionary file is constructed, the record number (that is, pointer) of the first entry (that is, value) for each character-type attribute is known. By using this information and the previously compiled list of names of attributes, the Mask file can be created. Next, all attributes should be grouped into categories of related information. This grouping provides the information necessary for the construction of the Definitions file. For multiple-choice-type attributes one simply needs to assign and to delineate value acronyms for each attribute in a record. Each group of value acronyms forms a set, and the collection of sets forms the Multiple-choice file.

Finally, the Numeric Master file is constructed, one record at a time. The individual record is constructed by assigning values for each attribute in the order of its occurrence in the Mask file. For integer or real attributes, the value is inserted directly. *For character-type attributes, the entry number of the value in the appropriate dictionary is inserted.*

For multiple-choice-type attributes, the binary encoded word that describes its value is inserted. The record is then compressed as described in the section "Data File Structures."

USE OF GRASP

From the viewpoint of a user, GRASP is a mechanism for obtaining information from a data bank in a very simplistic and rigid manner. The "language" which is used to "direct" GRASP is composed of 11 "commands." These commands can logically be divided into four groups.

GROUP 1 (FILES, NAMES) is used to:

A. Select or change the data base of interest.
B. Obtain information regarding the nomenclature and content of the selected data base.

GROUP 2 (LIST, DUMP, FUNCTION) is used to:

A. Examine a selected set of records that is called a file.
B. Perform selected computations of numeric attributes in a file.

GROUP 3 (CONDITIONS, LOGIC, SEARCH) is used to perform a retrieval (SEARCH) on the data bank based on given criteria (CONDITIONS) which are combined via a logical expression (LOGIC), a shorthand method of indicating which records of the data bank are to be retrieved.

GROUP 4 (HELP, REVIEW, QUIT) is used to:

A. Obtain brief information about the commands that GRASP will accept.
B. Obtain information regarding the history and status of the current session with GRASP.
C. Terminate the current session with GRASP.

All commands except HELP and REVIEW will ask for some type of response. Each response entered must end by striking the "cr" (RETURN) key. If a typing error or incorrect response is given, the system asks for another response. If at any point the system seems to be idle it is a good practice to strike the cr key. Certain commands (SEARCH, LIST, DUMP, FUNCTION) require an input file name. Entering a blank name in response to prompts generated by these commands (that is, cr only) results in the selection of the current Numeric Master file (as specified in the most recent FILE command). The LIST and DUMP commands also ask for the number of lines per page. This causes the system to pause after each printing of this number of lines, awaiting a response from the user. The user may then make a hard copy and (or) clear the screen if using a CRT terminal. Also, the user may terminate the printing altogether. At each pause, the user should enter a nonblank character followed by a cr if it is desired to abort the rest of the printout; otherwise, only a cr will continue printing. The method of calling the GRASP system into execution will vary, depending on what computer is used. At the beginning of execution, the GRASP system will print out the names and descriptions of the data bases available. The data-base name corresponds to the name of the Numeric Master file. Assume, for purposes of explanation, that a data consisting of oil and gas pools in the State of Colorado is available and named COLFIL. Following is a discussion of each command:

FILES.—This command is used to select a data base and may be issued at any time during a session. The individual-attribute names for any one data base will not be recognized by GRASP until this command has been issued. The user must enter a data-base name when the system asks for it.

NAMES.—This command is used to list the acronyms which will be used to identify individual attributes within a record (that is, pool) and their meaning. First, 17 categories are printed. Then the system asks the user to enter a list of numbers corresponding to the categories of interest. The list should be composed of individual numbers or number ranges (such as 2–5), each of which must be separated by a comma. The list must be terminated by the cr key: for example, 1, 2–5, 9 cr and 1–4, 10, 11 cr. The system then lists each acronym and its meaning for all the categories of interest. After each category is complete, the system pauses. At this point the user must enter cr to continue, or enter any letter (or digit) followed by cr to stop. After all categories have been completed, the system asks if the user would like to see the possible values of any character-type or multiple-choice-type items. The user must then enter Y or N followed by cr to indicate his decision. If the user enters N, the system will ask the user to enter his next command. If the user enters Y, the system asks for the names (acronyms) of the attributes of interest. The names are prompted and are given one per line followed by a cr. After each name is given, the system skips to the next line and prints a numeral. To end the list (a maximum of 10 names may appear), enter cr (with no name). The system then starts printing the attribute names and possible values, pausing after each name is complete. A pause also occurs after 30 lines of print. At each pause, enter cr to continue or any letter (or digit)

followed by cr to stop. After this process is completed, the system then asks if the user would like to see any more possible values. Again, enter Y for yes or N for no.

LIST.—This command is used to output selected attribute values (or expressions) from a selected file. Output may be to an interactive terminal or to a specified data set which could be processed at some later time by other programs. The system first asks the user for the input-file name and the number of lines per page. The user is next asked to enter C for column printing or R for row printing. If column output is selected, the user is asked if he wants output to be to a disk data set in character form. If so, the system will ask for a data-set name. Column output prints the selected acronyms as headings and their respective values below. Each column is composed of 8 character positions in a field of 10. One line of column output corresponds to one record. Row output consists of lines, each of which contain an acronym and its corresponding value. If the value for a selected attribute is missing, the attribute name is not printed. Records are separated by a line of asterisks. Before output proceeds, the system asks for the names of attributes or expressions which are desired. This is done by prompting with index numbers.

Expressions may optionally be preceded by some name. Five intrinsic functions are available: square root (SQRT), square (SQR), log base 10 (LOG), power of ten (TEN), and absolute value (ABS). Expressions may involve these intrinsic functions, attribute names, numeric constants, the arithmetic operators (+, −, *, /), and the grouping symbols ( ). The following is an example of a list to be output:

1. POOL
2. FIELD
3. DEPTH
4. LOG (DEPTH)
5. WELLAV=CRUAN69/(NUMPOOL-TOTPROD)
6.

In the above example, GRASP has prompted with the index numbers 1–6. Note that the list is terminated by a blank entry.

DUMP.—This command is very similar to the LIST command having row printing specified. Instead of asking for a list of names, the system asks for a list of category numbers. It then prints (in row fashion) the attribute name and value for each attribute present in the selected categories of the specified file.

FUNCTION.—This command performs functions on a file. Currently, the only functions available are the arithmetic mean (MEAN) and a linear least-square fit (FIT) of two attributes. The system asks for the name of the input file. Next, the user is asked for the name of the function and names of the arguments. The argument names are the acronyms for attributes within a record; as many as five may be given. For instance, if MEAN DEPTH, TOTPROD, CRUCM70 cr were entered, the range, mean, root mean square, sum, and sum of squares for DEPTH, TOTPROD, and CRUCM70 would be computed and printed. If FIT DEPTH, TOTPROD cr were entered, the system would respond with the slope, intercept, and correlation coefficient. Values for all attributes in a record must be present for that record to be included in a computation.

CONDITIONS.—This command is used to enter a set of retrieval criteria. Each criterion must be given in the form *acronym relation value,* where *acronym* is an attribute name (such as COUNTY, CRUCM69, POOL), where *relation* is EQ, NE, GT, LT, LE, GE, or BE, and where *value* is a number or a series of letters (such as ADAMS, 19342, MISSISSIPPIAN). The above relations have the following meanings:

EQ—equal to.
NE—not equal to.
GT—greater than.
LT—less than.
LE—less than or equal to.
GE—greater than or equal to.
BE—between (numerically, inclusive).

The system precedes each condition with a letter prompt (up to 26 may be entered), which will be used in the logic expression that combines the conditions. Entering cr by itself terminates the list of conditions. Following is an example of a set of conditions:

A. COUNTY	EQ	BACA
B. DEPTH	BE	5000,6000
C. TOTPROD	GE	10
D. LITHOL	NE	DOLOMITE
E. COUNTY	EQ	ADAMS
F. POOL	NE	MISSISSIPIAN
G.		

In the above example, the system provided the letters A through G as prompts.

LOGIC.—This command is used to enter a logical or connective expression which combines the pre-

viously entered conditions to form the retrieval criterion. The logical expression may be composed of the logical connectives (operators), the letters corresponding to the criteria entered via the CONDITIONS command, and the grouping symbols ( ). The logical connectives are AND, inclusive OR, and NOT (written .AND., .OR., .NOT.). Note that they are each bracketed by periods. Provision has also been made to use * (for AND), + (for OR), − (for NOT). Assume that the example conditions given in the preceding CONDITIONS command section had been entered. If the user wanted to retrieve the pools in Baca County that had a depth of 5,000–6,000 feet, the logic expression would be A .AND. B cr. If all the pools in Adams and Baca Counties except those of Mississippian age having dolomite lithology were desired, the logic expression would be (A .OR. E) .AND. (D .AND. F) cr. Note that the last pair of parentheses is not really needed. The ANDs will be applied before the ORs. NOTs are applied before ANDs and ORs. Hence, the first set (A. OR. E) is necessary so that the E is connected to A instead of to D. If one wanted to retrieve all pools with at least 10 producing wells having a depth greater than 6,000 feet or less than 5,000 feet, the logic expression would be .NOT. B .AND. C cr. If one wanted to retrieve all pools having less than 10 producing wells in the same range as above, one could use .NOT. (B .OR. C) cr for a logic expression. This expression, in words, says "if the pool has a depth of 5,000–6,000 feet, or if it has 10 or more producing wells, I don't want it."

SEARCH.—After the system has been given the conditions and connecting logic that compose the question to be asked of some file, an actual search of the data bank can be made. This is done with the SEARCH command. The system will ask for the name of the file to be searched (input file) and the name to call the file of records found (output file). After the search has been made, the system types the number of records searched and the number of records found. The capability of entering both input and output file names allows the user to perform "nested" searches. This means searches of files that were the result of previous searches. Frequently this is the most economical way of performing multiple or complex retrievals. For instance, suppose one wanted information on several sets of pools, all of which were in one county. One would first create an output file that contained all the pools in that county and then use that file as the input file for subsequent searches.

HELP.—This command is used to print a list of the possible commands and a brief description of their functions.

REVIEW.—This command provides a review of the conditions and logic which are currently in effect. The names of input and output files for the last 10 retrievals are also printed. This command is used to refresh one's memory on what was done recently during the current session.

QUIT.—This command is used to exit from the GRASP system. A list of the files created during this session is printed, and the user is permitted to save them for future use. Abnormal session interrupts and terminations will cause GRASP to cease functioning. However, all files created during the active session are either saved or not saved, according to the abnormal termination rules of the particular computing system. On abnormal termination, GRASP neither saves nor deletes files.

REFERENCES CITED

Bauer, F. L., and Samelson, K., 1960, Sequential formula translation: Assoc. Computing Machinery Commun., v. 2, no. 2, p. 76–83.

Bell, J. R., and Kaman, C. H., 1970, The linear quotient hash code: Assoc. Computing Machinery Commun., v. 13, no. 11, p. 675–677.

Lee, J. A. N., 1967, The anatomy of a compiler: New York, Reinhold Publishing Corp., 275 p.

GRASP SOFTWARE SPECIFICATIONS

MODULE NAME: DRIVER

Purpose: DRIVER is used primarily as a switching/calling mechanism. User commands are accepted and decoded. Control is then passed to the routine designed to process the given command. This process continues until the user "quits."

Subroutines called: START, KEYBRD, CONDTN, LOGEXP, RETRVE, FTNC, FILES, CONDS, HELP, DUMPIT, NAME, LIST, QUIT, PACK.

Common data referenced: None

Called by: None

Error checking and reporting: The command entered by the user is checked against the list of legal commands. If a command is not recognized, it is echoed back to the user terminal with a message suggesting use of the HELP command.

Program logic:

1. Initialization is performed by zeroing counters and calling START.
2. An unpacked character string is accepted from the user via subroutine KEYBRD.
3. A four-character command is formed by packing the above string into COMMAND.
4. COMMAND is then compared with the list (WORDS) of acceptable command words (NAMES). When a match is found, control is transferred to the appropriate subroutine via a computed GO TO.
5. Steps 2 through 4 are repeated until an end-of-file (EOF) condition is sensed on the terminal or until the QUIT command is executed.

G R A S P S O U R C E P R O G R A M

```
      INTEGER WORDS(11),COMAND,NAMEPT(26),RCODE(26),IVAL(26),POLISH(30)0000001
     1 ,IMAGE(5),IFILES(20),OFILES(20)                                0000002
      DATA WORDS/'COND','LOGI','SEAR','LIST','FILE','QUIT','NAME','HELP'0000003
     1,'REVI','DUMP','FUNC'/,IBLNK/' '/                               0000004
      NFILES=0                                                        0000005
      LPS=0                                                           0000006
      CALL START                                                      0000007
  110 TYPE 270                                                        0000008
      COMAND=IBLNK                                                    0000009
      CALL KEYBRD(&260,IMAGE,4)                                       0000010
      CALL PACK(IMAGE,COMAND,4,4)                                     0000011
      DO 120 I=1,11                                                   0000012
      IF (COMAND.EQ.WORDS(I)) GO TO 130                               0000013
  120 CONTINUE                                                        0000014
      TYPE 290, COMAND                                                0000015
      GO TO 110                                                       0000016
  130 GO TO (140,150,160,240,190,260,230,210,180,220,170), I          0000017
  140 CALL CONDTN(&110,NAMEPT,RCODE,IVAL,NREXP)                       0000018
      GO TO 110                                                       0000019
  150 CALL LOGEXP(&110,POLISH,LPS,NREXP)                              0000020
      GO TO 110                                                       0000021
  160 CALL RETRVE(&150,&110,IFILES,OFILES,NFILES,POLISH,LPS,NAMEPT,   0000022
     1 RCODE,IVAL,NREXP)                                              0000023
      GO TO 110                                                       0000024
  170 CALL FTNC(&110)                                                 0000025
      GO TO 110                                                       0000026
  190 CALL FILES(&110)                                                0000027
      GO TO 110                                                       0000028
  180 CALL CONDS(NREXP,LPS)                                           0000029
      IF (NFILES.GT.0) GO TO 200                                      0000030
      TYPE 300                                                        0000031
      GO TO 110                                                       0000032
  200 TYPE 310, (IFILES(I),OFILES(I),I=1,NFILES)                      0000033
      GO TO 110                                                       0000034
  210 CALL HELP(WORDS)                                                0000035
      GO TO 110                                                       0000036
  220 CALL DUMPIT                                                     0000037
      GO TO 110                                                       0000038
```

```
 230 CALL NAME(&110)                                                    0000039
     GO TO 110                                                          0000040
 240 CALL LIST(&110)                                                    0000041
     GO TO 110                                                          0000042
 260 CALL QUIT(OFILES,NFILES)                                           0000043
     STOP                                                               0000044
 270 FORMAT (//' ENTER COMMAND: ',$)                                    0000045
 290 FORMAT (1X,A5,' ILLEGAL COMMAND. ENTER HELP IF YOU WISH TO SEE',   0000046
    1' THE LEGAL COMMANDS.'/)                                           0000047
 300 FORMAT (' NO FILES HAVE BEEN USED AT THIS TIME.')                  0000048
 310 FORMAT (' INPUT:    OUTPUT:'/(2X,A5,2X,A5/))                       0000049
     END                                                                0000050
```

SUBROUTINE NAME: ACCESS

Purpose: ACCESS looks up character-string values in dictionary files. In order to minimize disk accesses, five previous values are saved for as many as 100 distinct character-type items.

Calling sequence: CALL ACCESS(NUMD,IVAL,TANK, NWORDS,ISWTCH)

Arguments:

NUMD—Item number of the character-type variable whose values are to be accessed.

IVAL—Direct-access key under which value is stored.

TANK—Contains the character value accessed.

NWORDS—The number of words in TANK.

ISWTCH—Switch to control which of the following four functions are desired:

1. Initialization for dictionary lookup.
2. Lookup a random item.
3. Return the direct-access key of the first item in this dictionary.
4. Return the indicated (by IVAL) entry and the KEY for the next entry (that is, reset IVAL).

Subroutines called: None

Common data referenced: IDPT in /DACOMM/

Called by: BDEF, COLPNT, DUMPIT, PNTER, ROWPNT

Error checking and reporting: None

Program Logic: The logic is divided into four sections relating to values of ISWTCH.

1. If ISWTCH=1, initialize saved pointer arrays (USED, LASTDX) and set character variable counter NCVAR to zero.
2. If ISWTCH=2, see if the value has been stored in BUFFER. If so, return it; otherwise access it on FORTRAN unit 21 and save its value (TANK), index (IVAL) and the item number (NUMD).
3. If ISWTCH=3, return the direct-access key for the first entry of the dictionary pointed to by NUMD.
4. If ISWTCH=4, access the entry pointed to by IVAL and reset IVAL to the key for the next entry in this dictionary.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE ACCESS(NUMD,IVAL,TANK,NWORDS,ISWTCH)                    0000051
      COMMON /DACOMM/ NV,IDPT                                            0000052
      INTEGER   TANK(1),BUFFER(5,100,5),USED(100),INDEX(100),            0000053
     1 LASTDX(5,100),IDPT(500)                                           0000054
      DATA IBLNK,NDICT/' ',21/                                           0000055
      GO TO (5,15,100,150),ISWTCH                                        0000056
    5 NCVAR=0                                                            0000057
      DO 10 J=1,100                                                      0000058
      USED(J)=0                                                          0000059
      DO 10 I=1,5                                                        0000060
   10 LASTDX(I,J)=-999999                                                0000061
      GO TO 320                                                          0000062
   15 IF(NCVAR.EQ.0) GO TO 30                                            0000063
      DO 20 KK=1,NCVAR                                                   0000064
      IF(INDEX(KK).EQ.NUMD) GO TO 40                                     0000065
   20 CONTINUE                                                           0000066
   30 NCVAR=NCVAR+1                                                      0000067
      IF(NCVAR.GT.100) NCVAR=100                                         0000068
      KK=NCVAR                                                           0000069
      INDEX(KK)=NUMD                                                     0000070
   40 IF(USED(KK).EQ.0) GO TO 240                                        0000071
      DO 50 K=1,5                                                        0000072
      IF(IVAL.EQ.LASTDX(K,KK)) GO TO 60                                  0000073
   50 CONTINUE                                                           0000074
      GO TO 240                                                          0000075
   60 NWORDS=5                                                           0000076
      DO 70 I=1,NWORDS                                                   0000077
   70 TANK(I)=BUFFER(I,KK,K)                                             0000078
      GO TO 320                                                          0000079
100     IVAL=IDPT(NUMD)                                                 0000080
      GO TO 320                                                          0000081
150     READ(NDICT'IVAL) NP,NWORDS,(TANK(I),I=1,NWORDS)                  0000082
      IVAL=NP                                                            0000083
      GO TO 320                                                          0000084
```

```
240      ISTART=IDPT(NUMD)                                                  0000085
         READ(NDICT'ISTART+IVAL-1) NP,NWORDS,(TANK(I),I=1,NWORDS)           0000086
         USED(KK)=MOD(USED(KK),5)+1                                         0000087
         NUSED=USED(KK)                                                     0000088
         DO 260 I=1,5                                                       0000089
  260    BUFFER(I,KK,NUSED)=IBLNK                                           0000090
         NWORD=MINO(NWORDS,5)                                               0000091
         DO 270 I=1,NWORD                                                   0000092
  270    BUFFER(I,KK,NUSED)=TANK(I)                                         0000093
         LASTDX(NUSED,KK)=IVAL                                              0000094
  320    RETURN                                                             0000095
         END                                                                0000096
```

SUBROUTINE NAME: BDEF

Purpose: BDEF provides access to the character and multiple-choice dictionaries.

Calling sequence: CALL BDEF(&n)

Argument:

n—Statement number (in caller) to which a branch is made if an EOF is sensed by KEYBRD.

Subroutines called: KEYBRD, VLIST, ACCESS, PAUSE, BINTYP, IFILE

Common data referenced:

FNAMES, WHICH in /FILNAM/
ITYPE in blank common

Called by: NAME

Error checking and reporting: The user response to a "yes/no" question is checked. If illegal, a message is typed, and the user is prompted for another response. If an item is selected that is not a multiple-choice or character-type item, a message is typed, and the user is requested to reenter a list of item names.

Program logic: The user is prompted to determine if a list of multiple-choice or character-type values is desired. If the response (obtained from KEYBRD) is "N", a branch is made to the end of the routine. If the response is "Y," a list of item names is obtained by a call to VLIST. If an EOF is sensed, the nonstandard return (from VLIST) exits via the nonstandard return of BDEF. The data type is determined for each name (TAGS) returned by VLIST. If the type is not multiple choice or character, that message is typed, and the next element of TAGS is considered. If the type is multiple choice, a call to BINTYP is used to obtain the permissible values for printing. If the type is character, calls to ACCESS are made to obtain the possible values. When the second argument of ACCESS is returned as zero, all possible values have been referenced. A programmed pause is generated after each 30 lines of print and after each item in TAGS is processed. Just prior to return (standard or nonstandard), unit 22 (the binary or multiple-choice file) is rewound, and the unit number is reassociated with the current file name via a call to IFILE.

GRASP SOURCE PROGRAM

```
      SUBROUTINE BDEF(*)                                                        0000097
      COMMON NAMES,ITYPE,IPTS,IPAD                                              0000098
      COMMON /FILNAM/ FNAMES,WHICH,PAD                                          0000099
      DOUBLE PRECISION LABEL(25),NAMES(500),TAGS(20)                            0000100
      INTEGER IPTS(500),FNAMES(21),WHICH,PAD(4)                                 0000101
      INTEGER BINL(20),YES,NO,REPLY,BITEM(15,25),ITYPE(500),TANK(25)            0000102
      EQUIVALENCE (BITEM(1,1),TANK(1))                                          0000103
      DATA YES,NO,NBIN/'Y','N',22/                                              0000104
   10 TYPE 120                                                                  0000105
   20 TYPE 130                                                                  0000106
      CALL KEYBRD(&110,REPLY,1)                                                 0000107
      IF (REPLY.EQ.NO) GO TO 100                                                0000108
      IF (REPLY.EQ.YES) GO TO 30                                                0000109
      TYPE 150                                                                  0000110
      GO TO 20                                                                  0000111
   30 CALL VLIST(&110,TAGS,BINL,NUM)                                            0000112
      DO 90 N=1,NUM                                                             0000113
      INDEX=BINL(N)                                                             0000114
      IF (ITYPE(INDEX)-3) 60,40,70                                              0000115
   40 TYPE 190, TAGS(N)                                                         0000116
      CALL ACCESS(INDEX,K,TANK,NOM,3)                                           0000117
      DO 50 J=1,10000                                                           0000118
      CALL ACCESS(INDEX,K,TANK,M,4)                                             0000119
      TYPE 160, (TANK(I),I=1,M)                                                 0000120
      IF(K.EQ.0) GO TO 90                                                       0000121
      IF (MOD(J,30).NE.0) GO TO 50                                              0000122
      CALL PAUSE(&100)                                                          0000123
   50 CONTINUE                                                                  0000124
   60 TYPE 170, TAGS(N)                                                         0000125
      GO TO 90                                                                  0000126
   70 IF(ITYPE(INDEX).NE.4) GO TO 60                                            0000127
      CALL BINTYP(INDEX,LABEL,BITEM,K,M)                                        0000128
      TYPE 190, TAGS(N)                                                         0000129
      DO 80 J=1,M                                                               0000130
   80 TYPE 180, LABEL(J),(BITEM(I,J),I=1,K)                                     0000131
```

```
   90 CALL PAUSE(&100)                                                    0000132
      GO TO 10                                                            0000133
100     REWIND NBIN                                                      0000134
      CALL IFILE(NBIN,FNAMES(16+WHICH))                                   0000135
      RETURN                                                              0000136
110     REWIND NBIN                                                      0000137
      CALL IFILE(NBIN,FNAMES(16+WHICH))                                   0000138
      RETURN 1                                                            0000139
  120 FORMAT (' WOULD YOU LIKE TO SEE THE POSSIBLE VALUES ','OF MULTIPLE0000140
     1 CHOICE'/' OR CHARACTER TYPE ITEMS?')                               0000141
  130 FORMAT ('  (ENTER Y FOR YES , N FOR NO): ',$)                       0000142
  150 FORMAT (' YOUR REPLY WAS NOT UNDERSTOOD.')                          0000143
  160 FORMAT (10X,12A5)                                                   0000144
  170 FORMAT (1X,A8,'IS NOT A MULTIPLE CHOICE OR CHARACTER TYPE ITEM.')   0000145
  180 FORMAT (11X,A7,'-',15A4)                                            0000146
  190 FORMAT (1X,A8,'=')                                                  0000147
      END                                                                 0000148
```

SUBROUTINE NAME: BFIND

Purpose: BFIND is used to look up a double-word-item name in a list of double-word-item names having a given sorted order.

Calling sequence: CALL BFIND(&n,KEY,IPOST,KEYS, INDEX,K)

Arguments:

n—Statement number (in caller) to which a branch is made if the name KEY is not in the list of names KEYS.

KEY—The double-word item to be looked up.

IPOST—The position in KEYS of the item KEY.

KEYS—The list of double-word names that will be used to look up KEY.

INDEX—Gives the indices of the sorted form of KEYS.

K—The number of elements in KEYS.

Subroutines called: None

Common data referenced: None

Called by: DECOMP, PARSE, RELEXP, VLIST

Error checking and reporting: None

Program logic: The standard binary-search technique is used, which repeatedly halves the interval of search on a sorted list. If the interval of search goes negative, the element is not in the list and the nonstandard return is taken.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE BFIND(*,KEY,IPOST,KEYS,INDEX,K)                    0000149
      DOUBLE PRECISION KEY,KEYS(1)                                  0000150
      DIMENSION INDEX(1)                                            0000151
      L1=1                                                          0000152
      L2=K                                                          0000153
   70 IF (L2.LT.L1) RETURN 1                                        0000154
      J=(L1+L2)/2                                                   0000155
      I=INDEX(J)                                                    0000156
      IF (DABS(KEY).GT.DABS(KEYS(I))) GO TO 90                      0000157
      IF (DABS(KEY).LT.DABS(KEYS(I))) GO TO 80                      0000158
      GO TO 100                                                     0000159
   80 L2=J-1                                                        0000160
      GO TO 70                                                      0000161
   90 L1=J+1                                                        0000162
      GO TO 70                                                      0000163
  100 IPOST=I                                                       0000164
      RETURN                                                        0000165
      END                                                           0000166
```

SUBROUTINE NAME: BINIT

Purpose: BINIT is used to sort a list of double-precision words (NAMES, in this instance) into ascending order and return an array of indices giving the sorted order of the elements in terms of the unsorted order.

Calling sequence: CALL BINIT(KEYS,INDEX,M)

Arguments:

KEYS—The list of double-precision words to be sorted.

INDEX—An array of indicies giving the unsorted order of KEYS.

M—The number of words in the KEYS and INDEX arrays.

Subroutines called: None

Common data referenced: None

Called by: FILES

Error checking and reporting: None

Program logic: An in-place sort is performed using the standard "Shell" technique. The original order of KEYS is overwritten, and the sorted order is returned in INDEX.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE BINIT(KEYS,INDEX,M)                                    0000167
      DOUBLE PRECISION KEYS(1),KTEMP                                    0000168
      INTEGER INDEX(1)                                                  0000169
      DO 10 J=1,M                                                       0000170
   10 INDEX(J)=J                                                        0000171
      MO=M                                                              0000172
   20 IF (MO.LE.1) GO TO 60                                             0000173
      J=4                                                               0000174
      IF (MO.GT.15) J=8                                                 0000175
      MO=2*(MO/J)+1                                                     0000176
      KO=M-MO                                                           0000177
      JO=1                                                              0000178
   30 I=JO                                                              0000179
   40 J=I+MO                                                            0000180
      IF (DABS(KEYS(I)).LE.DABS(KEYS(J))) GO TO 50                      0000181
      KTEMP=KEYS(I)                                                     0000182
      KEYS(I)=KEYS(J)                                                   0000183
      KEYS(J)=KTEMP                                                     0000184
      ITEMP=INDEX(I)                                                    0000185
      INDEX(I)=INDEX(J)                                                 0000186
      INDEX(J)=ITEMP                                                    0000187
      I=I-MO                                                            0000188
      IF (I-1) 50,40,40                                                 0000189
   50 JO=JO+1                                                           0000190
      IF (JO-KO) 30,30,20                                               0000191
   60 RETURN                                                            0000192
      END                                                               0000193
```

SUBROUTINE NAME: BINTYP

Purpose: BINTYP reads the values (LABEL, BITEM) which can be assumed by the multiple-choice-type item whose number is NUMD.

Calling sequence: CALL BINTYP,(NUMD,LABEL,BITEM, LNG,NUM)

Arguments:

NUMD—Integer specifying a multiple-choice-type item.

LABEL—Set of values (double word) that may be assumed by this item.

BITEM—Set of descriptions corresponding to elements of LABEL.

LNG—Maximum length of a description.

NUM—The number of descriptions.

Subroutines called: IFILE

Common data referenced: FNAMES, WHICH in /FILNAM/

Called by: BDEF, COLPNT, DUMPIT, PNTER, ROWPNT

Error checking and reporting: None

Program logic: Records of the multiple-choice (binary) file are read sequentially until the correct record is obtained. If the file was positioned past the desired record at call time, an EOF is sensed, the file is rewound, and the current multiple-choice file name is reassociated with unit 22.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE BINTYP(NUMD,LABEL,BITEM,LNG,NUM)                    0000194
      COMMON /FILNAM/ FNAMES,WHICH,PAD                               0000195
      DOUBLE PRECISION LABEL(1)                                      0000196
      INTEGER FNAMES(21),WHICH,PAD(4),BITEM(15,25)                   0000197
      DATA NBIN/22/                                                  0000198
  300 READ(NBIN,END=310) NAME,NUM,LNG,                               0000199
     A (LABEL(J),(BITEM(I,J),I=1,LNG),J=1,NUM)                       0000200
      IF(NUMD-NAME) 310,320,300                                      0000201
  310 REWIND NBIN                                                    0000202
      CALL IFILE(NBIN,FNAMES(16+WHICH))                              0000203
      GO TO 300                                                      0000204
  320 RETURN                                                         0000205
      END                                                            0000206
```

SUBROUTINE NAME: BLIST

Purpose: BLIST returns a list of numbers giving the bit positions of the "ones" in a binary word.

Calling sequence: CALL BLIST(LIST,NUML,ICODE)

Arguments:

LIST—Array of integers giving the bit positions in ICODE which are "ones," counting right to left.

NUML—The number of items in LIST.

ICODE—The binary word to be examined by BLIST.

Subroutines called: None

Common data referenced: None

Called by: COLPNT, DUMPIT, ROWPNT

Error checking and reporting: None

Program logic: ICODE is moved to IDUM. IDUM is successively divided by 2, and the least significant bit is accessed by the MOD function. If the least significant bit is "one," the position counter is added to LIST.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE BLIST(LIST,NUML,ICODE)                              0000207
      INTEGER LIST(1)                                                0000208
      NUML=0                                                         0000209
      IDUM=ICODE                                                     0000210
      DO 10 I=1,25                                                   0000211
      IF (IDUM.EQ.0) GO TO 20                                        0000212
      IF (MOD(IDUM,2).EQ.0) GO TO 10                                 0000213
      NUML=NUML+1                                                    0000214
      LIST(NUML)=I                                                   0000215
   10 IDUM=IDUM/2                                                    0000216
   20 RETURN                                                         0000217
      END                                                            0000218
```

SUBROUTINE NAME: COLPNT

Purpose: COLPNT outputs the values of as many as 20 selected items or expressions. Output is columnar and is directed to the terminal or to a disk data set.

Calling sequence: CALL COLPNT(&n,NPAGE)

Arguments:

n—Statement (in calling routine) to which a branch is made if the nonstandard return is taken from VLIST (KEYBRD senses EOF).

NPAGE—Number of lines between pauses per page of terminal output.

Subroutines called: KEYBRD, OFILE, VLIST, ACCESS, GETPUT, PAUSE, EVAL, UNCODE, BINTYP, BLIST, PACK

Common data referenced:

POLISH, ICODE, LPS in/EXPRNS/
ITYPE in blank common

Called by: LIST

Error checking and reporting: None

Program logic:

1. The user is asked if he would like the output to go to disk. His reply is returned by KEYBRD. If affirmative a logical flag is set, and he is prompted for a data-set name. This name is then associated with unit 24 via a call to OFILE.
2. A call to VLIST returns the item names (or expression pointers) that are selected.
3. A call to ACCESS is made to initialize the lookup of character dictionary values.
4. Each record of the selected file is then obtained via GETPUT, and a line (or record) of output is constructed. For numeric data, a format is constructed to maximize the number of significant digits displayed, and the constructed line is printed.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE COLPNT(*,NPAGE)                                          0000219
      COMMON NAMES,ITYPE,IPTS,IDIM                                        0000220
      COMMON /EXPRNS/ POLISH,ICODE,LPS                                    0000221
      DIMENSION ITYPE(500),BITEM(15,25),ITEMS(20),IREC(500),IPTS(500),    0000222
     1REC(500), NAMES(500), TANK(25), LABEL(25), LIST(25), POLISH(        0000223
     215,8), ICODE(15,8), LPS(8), IQUAL(6)                                0000224
      DOUBLE PRECISION DBLNK,AREA,LINE(20),NAMES,LABEL,VNAMES(20),        0000225
     1 FMT(3),FMTS(8)                                                     0000226
      INTEGER BLANK,TANK,YES                                              0000227
      LOGICAL BLNK,TTY                                                    0000228
      EQUIVALENCE (REC,IREC),(IVAL,VAL),(TANK,LIST),(BLANK,IQUAL)         0000229
      DATA FMT,FMTS/'(     ','      ',',A1)  ','F8.6  ','F8.5  ',       0000230
     1 'F8.4  ','F8.3  ','F8.2  ','F8.1  ','F8.0  ','1PE8.1'/,           0000231
     2 DBLNK,YES,IQUAL/'      ','Y',' ','G','H','L','N','T'/             0000232
      TYPE 100                                                            0000233
      CALL KEYBRD(&290,I,1)                                               0000234
      TTY=I.NE.YES                                                        0000235
      IF(TTY) GO TO 115                                                   0000236
      NPAGE=10000000                                                      0000237
      TYPE 105                                                            0000238
      CALL KEYBRD(&290,ITEMS,5)                                           0000239
      I=BLANK                                                             0000240
      CALL PACK(ITEMS,I,5,5)                                              0000241
      CALL OFILE(24,I)                                                    0000242
115     KOUNT=0                                                           0000243
      CALL VLIST(&290,VNAMES,ITEMS,NUM)                                   0000244
      IF (NUM.EQ.0) GO TO 280                                             0000245
      IF(TTY) TYPE 300, (VNAMES(I),I=1,NUM)                               0000246
      CALL ACCESS(II,IVAL,TANK,LK,1)                                      0000247
  120 CALL GETPUT(&270,IREC,1)                                            0000248
      KOUNT=KOUNT+1                                                       0000249
      IF (KOUNT.LE.NPAGE) GO TO 130                                       0000250
      KOUNT=0                                                             0000251
      CALL PAUSE(&270)                                                    0000252
      TYPE 300, (VNAMES(I),I=1,NUM)                                       0000253
```

```
  130 DO 260 JJ=1,NUM                                                    0000254
      AREA=DBLNK                                                         0000255
      II=ITEMS(JJ)                                                       0000256
      IF (II.GT.0) GO TO 140                                             0000257
      II=-II                                                             0000258
      VAL=EVAL(IREC,ICODE(1,II),POLISH(1,II),LPS(II),BLNK)               0000259
      IF(BLNK) GO TO 260                                                 0000260
      GO TO 160                                                          0000261
  140 IVAL=IREC(II)                                                      0000262
      IF (IVAL.EQ.BLANK) GO TO 260                                       0000263
      KIND=ITYPE(II)                                                     0000264
      GO TO (150,160,210,250,205), KIND                                  0000265
  150 ENCODE(8,320,AREA) IVAL                                            0000266
      GO TO 260                                                          0000267
  160 IF (VAL.EQ.0.) GO TO 180                                           0000268
      A=ALOG10(ABS(VAL))                                                 0000269
      IF (A.GE.5.) GO TO 190                                             0000270
      IF (A.LE.-4.) GO TO 190                                            0000271
      IF (A.LE.0.) GO TO 170                                             0000272
      LK=IFIX(A)+2                                                       0000273
      GO TO 200                                                          0000274
  170 LK=1                                                               0000275
      GO TO 200                                                          0000276
  180 LK=2                                                               0000277
      GO TO 200                                                          0000278
  190 LK=8                                                               0000279
  200 FMT(2)=FMTS(LK)                                                    0000280
      IF(KIND.NE.5) IQ=1                                                 0000281
      ENCODE(9,FMT,AREA)  VAL,IQUAL(IQ)                                  0000282
      GO TO 260                                                          0000283
  205 VAL=UNCODE(VAL,IQ)                                                 0000284
      GO TO 160                                                          0000285
  210 CALL ACCESS(II,IVAL,TANK,LK,2)                                     0000286
      ENCODE(8,215,AREA) (TANK(I),I=1,LK)                                0000287
      GO TO 260                                                          0000288
  250 CALL BINTYP(II,LABEL,BITEM,K,M)                                    0000289
      CALL BLIST(LIST,NUMS,IVAL)                                         0000290
      AREA=LABEL(LIST(1))                                                0000291
  260 LINE(JJ)=AREA                                                      0000292
      IF(TTY) GO TO 65                                                   0000293
      WRITE(24,310) (LINE(JJ),JJ=1,NUM)                                  0000294
      GO TO 120                                                          0000295
65      TYPE 311,(LINE(JJ),JJ=1,NUM)                                   0000296
      GO TO 120                                                          0000297
  270 CONTINUE                                                           0000298
280     IF(.NOT.TTY) REWIND 24                                           0000299
      RETURN                                                             0000300
  290 RETURN 1                                                           0000301
100     FORMAT(' WOULD YOU LIKE OUTPUT TO BE TO DISK? (Y OR N): ',$)     0000302
105     FORMAT(' ENTER NAME OF DISK DATA SET TO BE CREATED: ',$)         0000303
215     FORMAT(A5,A3)                                                    0000304
  300 FORMAT (///1X,8A10)                                                0000305
  310 FORMAT (20A10)                                                     0000306
311     FORMAT(1X,8A10)                                                  0000307
  320 FORMAT (I8)                                                        0000308
      END                                                                0000309
```

FUNCTION NAME: COMP

Purpose: COMP evaluates a relational expression. The relational operators may be (in FORTRAN notation) EQ, LT, GT, LE, GE, NE, or BE (denoting between). The evaluation is performed on integer, real, or binary—(that is, bit) type data.

Calling sequence: HIT = COMP(IVAR,IVAL,VAR,VAL, ICODE,ISWTCH), where HIT is of type LOGICAL

Arguments:

IVAR, IVAL—Integer-type arguments to be compared if ISWTCH is 1.

VAR, VAL—Real-type arguments to be compared if ISWTCH is 2.

ICODE—Encoding of comparison to be made. Assumes values 1 through 7, respectively, indicating the relations EQ, LT, GT, LE, GE, NE, BE.

ISWTCH—Indicates the type of arguments for the comparison (integer, real, or binary).

Subroutines called: None

Common data referenced: IVALS in /BTWN/

Called by: RETRVE

Error checking and reporting: If ISWTCH is 3 (that is, binary-type arguments), ICODE is tested for 1 or 6 (EQ or NE). If the test fails, a message is typed and COMP returns FALSE as a value.

Program logic:

1. The logical variable NONBLK is set to indicate nonblank operands.
2. If ISWTCH is less than 3, the two operands are compared according to ICODE. If ICODE is 7 (indicating the between operator), the operands are obtained from IVALS in /BTWN/ by use of IVAL as a pointer to the appropriate set.
3. If ISWTCH is 3, a bit (binary) comparison is made by examining the IVAL'th bit in IVAR.

G R A S P S O U R C E P R O G R A M

```
      LOGICAL FUNCTION COMP(IVAR,IVAL,VAR,VAL,ICODE,ISWTCH)                          0000310
      COMMON /BTWN/ IVALS,NBE                                                        0000311
      DIMENSION IVALS(2,10), VALS(2,10)                                              0000312
      LOGICAL COMPAR,NONBLK                                                          0000313
      EQUIVALENCE (IVALS(1,1),VALS(1,1)), (I,VI), (BLANK,IBLNK)                      0000314
      DATA BLANK/' '/                                                                0000315
      NONBLK=IVAR.NE.IBLNK.AND.IVAL.NE.IBLNK                                         0000316
      IF(ISWTCH-2) 5,85,165                                                          0000317
    5 GO TO (10,20,30,40,50,60,70), ICODE                                            0000318
   10 COMPAR=IVAR.EQ.IVAL                                                          0000319
      GO TO 160                                                                      0000320
   20 COMPAR=IVAR.LT.IVAL.AND.NONBLK                                               0000321
      GO TO 160                                                                      0000322
   30 COMPAR=IVAR.GT.IVAL.AND.NONBLK                                               0000323
      GO TO 160                                                                      0000324
   40 COMPAR=IVAR.LE.IVAL.AND.NONBLK                                               0000325
      GO TO 160                                                                      0000326
   50 COMPAR=IVAR.GE.IVAL.AND.NONBLK                                               0000327
      GO TO 160                                                                      0000328
   60 COMPAR=IVAR.NE.IVAL                                                          0000329
      GO TO 160                                                                      0000330
   70 COMPAR=IVAR.GE.IVALS(1,IVAL).AND.IVAR.LE.IVALS(2,IVAL).AND.NONBLK            0000331
      GO TO 160                                                                      0000332
   85 NONBLK=VAR.NE.BLANK.AND.VAL.NE.BLANK                                         0000333
      GO TO (90,100,110,120,130,140,150), ICODE                                      0000334
   90 COMPAR=VAR.EQ.VAL                                                            0000335
      GO TO 160                                                                      0000336
  100 COMPAR=VAR.LT.VAL.AND.NONBLK                                                 0000337
      GO TO 160                                                                      0000338
  110 COMPAR=VAR.GT.VAL.AND.NONBLK                                                 0000339
      GO TO 160                                                                      0000340
  120 COMPAR=VAR.LE.VAL.AND.NONBLK                                                 0000341
      GO TO 160                                                                      0000342
  130 COMPAR=VAR.GE.VAL.AND.NONBLK                                                 0000343
      GO TO 160                                                                      0000344
  140 COMPAR=VAR.NE.VAL                                                            0000345
      GO TO 160                                                                      0000346
```

```
150 VI=VAL                                                                  0000347
    COMPAR=VAR.GE.VALS(1,I).AND.VAR.LE.VALS(2,I).AND.NONBLK                 0000348
160  COMP=COMPAR                                                            0000349
    GO TO 180                                                               0000350
165 IF(NONBLK) GO TO 170                                                    0000351
    IF(ICODE.EQ.1) GO TO 10                                                 0000352
    IF(ICODE.EQ.6) GO TO 60                                                 0000353
    TYPE 200                                                                0000354
     COMP=.FALSE.                                                           0000355
    GO TO 180                                                               0000356
170 IDIGIT=IVAR/2**(IVAL-1)                                                 0000357
    IF (ICODE.NE.1) GO TO 190                                               0000358
     COMP=MOD(IDIGIT,2).EQ.1                                                0000359
    GO TO 180                                                               0000360
190 IF (ICODE.NE.6) TYPE 200                                                0000361
     COMP=MOD(IDIGIT,2).NE.1                                                0000362
180 RETURN                                                                  0000363
200 FORMAT (52H RELATION MUST BE .EQ./.NE. FOR BINARY TYPE VARIABLE)        0000364
    END                                                                     0000365
```

SUBROUTINE NAME: CONDS

Purpose: CONDS types out the last set of conditions and logic entered.

Calling sequence: CALL CONDS(NCONDS,LPS)

Arguments:

NCONDS—The number of conditions currently entered.

LPS—The length of the current logical expression.

Common data referenced: EXPR, LOGIC in /INPUT/

Called by: DRIVER

Error checking and reporting: None

Program logic:

1. If any conditions have been entered (that is, NCOND >0), they are typed out.
2. If a logic expression has been entered (that is, LPS >0), it is typed out.

GRASP SOURCE PROGRAM

```
      SUBROUTINE CONDS(NCOND,LPS)                                          0000366
      COMMON /INPUT/ EXPR,LOGIC                                            0000367
      DIMENSION LABELS(26)                                                 0000368
      DOUBLE PRECISION EXPR(4,26),LOGIC(8)                                 0000369
      DATA LABELS/'A','B','C','D','E','F','G','H','I','J','K','L','M','N0000370
     1','O','P','Q','R','S','T','U','V','W','X','Y','Z'/                   0000371
      TYPE 40                                                              0000372
      IF (NCOND.EQ.0) GO TO 20                                             0000373
      DO 10 J=1,NCOND                                                      0000374
   10 TYPE 50, LABELS(J),(EXPR(I,J),I=1,4)                                 0000375
      GO TO 30                                                             0000376
   20 TYPE 60                                                              0000377
   30 IF (LPS.NE.0) TYPE 70, LOGIC                                         0000378
      RETURN                                                               0000379
   40 FORMAT (//)                                                          0000380
   50 FORMAT (4X,A1,'.  ',4A10)                                            0000381
   60 FORMAT (' NO CONDITIONS HAVE BEEN ENTERED YET.')                     0000382
   70 FORMAT (' LOGIC STATEMENT IS: ',8A10)                                0000383
      END                                                                  0000384
```

SUBROUTINE NAME: CONDTN

Purpose: This subroutine is used to control the entry of "conditions" (see RELEXP). Conditions are entered via KEYBRD, decoded via RELEXP, and saved (in character form) in the common area named INPUT.

Calling sequence: CALL CONDTN(&n,VARS,CODES,VALS, NCOND)

Arguments:

n—Statement number (in caller) to which a branch will be made on a nonstandard return.

VARS—Array of pointers to items referenced in the set of entered conditions.

CODES—Array of integers giving the individual relational operators in the entered conditions.

VALS—Array of values associated with the conditions entered.

NCOND—Counter giving the number of conditions entered.

Subroutines called: KEYBRD, RELEXP, PACK

Common data referenced:

NBE in /BTWN/
EXPR in /INPUT/

Called by: DRIVER

Error checking and reporting: An error flag returned by RELEXP is tested. If set, a request to retype the condition is issued.

Program logic:

1. A letter (A–Z) is printed as a prompt.
2. The user response (a condition) is obtained via KEYBRD and passed to RELEXP.
3. If no errors have been detected by RELEXP, the user response is packed into AREA and then moved (first 40 characters) into the INPUT common block.
4. Steps 1–3 are repeated and incrementally counted until an all-blank response is entered. When this occurs, execution resumes at statement 40, which sets the number of conditions entered. Control returns to the caller.
5. The nonstandard return is taken if an EOF is sensed from the terminal.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE CONDTN(*,VARS,CODES,VALS,NCOND)                        0000385
      COMMON /BTWN/ IVALS,NBE                                           0000386
      COMMON /INPUT/ EXPR,LOGIC                                         0000387
      DOUBLE PRECISION EXPR(4,26),AREA(4),LOGIC(8)                      0000388
      INTEGER IMAGE(80),VARS(1),VALS(1),CODES(1),IVALS(2,10),PROMPT(26) 0000389
      LOGICAL ERR                                                       0000390
      DATA PROMPT/'A.','B.','C.','D.','E.','F.','G.','H.','I.','J.','K.'0000391
     1,'L.','M.','N.','O.','P.','Q.','R.','S.','T.','U.','V.','W.','X.',0000392
     2'Y.','Z.'/                                                        0000393
      NBE=0                                                             0000394
      J=1                                                               0000395
   10 TYPE 50, PROMPT(J)                                                0000396
      CALL KEYBRD(&45,IMAGE,80)                                         0000397
      CALL RELEXP(&40,IMAGE,VARS(J),CODES(J),VALS(J),ERR)               0000398
      IF (ERR) GO TO 30                                                 0000399
      CALL PACK(IMAGE,AREA,40,40)                                       0000400
      DO 20 I=1,4                                                       0000401
   20 EXPR(I,J)=AREA(I)                                                 0000402
      J=J+1                                                             0000403
      GO TO 10                                                          0000404
   30 TYPE 60                                                           0000405
      GO TO 10                                                          0000406
   40 NCOND=J-1                                                         0000407
      RETURN                                                            0000408
   45 RETURN 1                                                          0000409
   50 FORMAT (1X,A3,$)                                                  0000410
   60 FORMAT (' RE-TYPE CONDITION')                                     0000411
      END                                                               0000412
```

SUBROUTINE NAME: DECOMP

Purpose: DECOMP extracts a list of item names and a corresponding list of item numbers from an unpacked character string.

Callng sequence: CALL DECOMP(&l,&m,IMAGE,NLIST, LIST,N)

Arguments:

l—Statement (in caller) which will be branched to if an invalid item name is detected.

m—Statement (in caller) which will be branched to if the input string contains no item names.

IMAGE—Unpacked input-character string.

NLIST—List of item names (offset by one from LIST).

LIST—A function name followed by a list of item numbers.

N—Total number of items in LIST.

Subroutines called: BFIND, PACK

Common data referenced: NAMES, IPTS, IDIM in blank common

Called by: FTNC

Error checking and reporting: BFIND takes the nonstandard return if a name is not found. This causes an error message to be typed and a new input to be requested. If a comma is detected before the list of item names begins, a message is typed and new input is requested.

Program logic: The input-string IMAGE is scanned, a character at a time, via a transition matrix. The list of names is created and the list of item numbers is obtained via calls to BFIND. The following transition matrix is used:

IMAT (4, 3)

	blank	comma	nonblank	purpose
1	f(0)/1	error	f(1)/2	start function name
2	f(2)/3	error	f(0)/2	find end of function name
3	f(0)/3	f(0)/3	f(3)/4	start item name
4	f(2)/3	f(2)/3	f(0)/4	find end of item name

where the f(i) are:

f(0)—No operation.

f(1)—Mark first character.

f(2)—Mark last character. Pack and find index of item name (that is, find item number).

f(3)—Increment list item counter and mark first character of new list item.

The entire input stream is scanned, and control is returned to caller. Refer to program logic section of LOGEXP for a more complete discussion of transition-matrix parsing.

GRASP SOURCE PROGRAM

```
      SUBROUTINE DECOMP(*,*,IMAGE,NLIST,LIST,N)                          0000413
      COMMON NAMES,ITYPE,IPTS,IDIM                                       0000414
      DIMENSION ITYPE(500), IMAGE( 1), LIST(1), IPTS(500), IMAT(4,3)     0000415
      DOUBLE PRECISION NAMES(500),NAME,DBLNK,NLIST(5)                    0000416
      EQUIVALENCE (INAME,NAME)                                           0000417
      DATA DBLNK,IBLNK,ICOMMA/'      ',' ',','/                         0000418
      DATA IMAT/1,23,3,23,2*40,3,23,12,2,34,4/                           0000419
      N=1                                                                0000420
      IROW=1                                                             0000421
      DO 90 I=1,80                                                       0000422
      IF (IMAGE(I).EQ.ICOMMA) GO TO 10                                   0000423
      IF (IMAGE(I).NE.IBLNK) GO TO 20                                    0000424
      IVAL=IMAT(IROW,1)                                                  0000425
      GO TO 30                                                           0000426
   10 IVAL=IMAT(IROW,2)                                                  0000427
      GO TO 30                                                           0000428
   20 IVAL=IMAT(IROW,3)                                                  0000429
   30 IROW=MOD(IVAL,10)                                                  0000430
      JOB=IVAL/10+1                                                      0000431
      GO TO (90,50,70,40,120), JOB                                       0000432
   40 N=N+1                                                              0000433
   50 IC=I                                                               0000434
      GO TO 90                                                           0000435
   70 LC=I-1                                                             0000436
      NAME=DBLNK                                                         0000437
      CALL PACK(IMAGE(IC),NAME,LC-IC+1,8)                                0000438
      IF (N.GT.1) GO TO 80                                               0000439
      LIST(1)=INAME                                                      0000440
      GO TO 90                                                           0000441
   80 CALL BFIND(&100,NAME,LIST(N),NAMES,IPTS,IDIM)                      0000442
      NLIST(N-1)=NAME                                                    0000443
```

```
 90 CONTINUE                                                              0000444
    IF (N.EQ.1) RETURN 2                                                  0000445
    RETURN                                                                0000446
100 TYPE 140, NAME                                                        0000447
110 RETURN 1                                                              0000448
120 TYPE 150, IMAGE(I)                                                    0000449
    GO TO 110                                                             0000450
140 FORMAT (1X,A8,'IS AN INVALID NAME. RE-ENTER LINE.')                   0000451
150 FORMAT (' PUNCTUATION ERROR CAUSED BY ',A1,'. RE-ENTER LINE.')        0000452
    END                                                                   0000453
```

SUBROUTINE NAME: DEFINE

Purpose: DEFINE is used to define the structure and name of a direct-access disk-data set having fixed-length records. Individual records may then be directly accessed by specifying the record number.

Calling sequence: CALL DEFINE (U,S,V,F,PJ,PG)

Arguments:

U—The FORTRAN unit number expressed as an integer.

S—The size of the records within the file, expressed as an integer. For formatted records, S gives the number of characters per record. For unformatted records, S gives the number of words per record.

V—The associated integer variable. The record number which would be accessed next if I/O were to continue sequentially is returned as an integer in the associated variable after each random read or write.

F—The name of the file which will be accessed when an I/O statement references U (the above unit number).

PJ—The project number in octal of the disk area in which the file resides.

PG—The programmer number in octal of the disk area in which the file resides.

Subroutines called: None

Common data referenced: None

Called by: FILES

Error checking and reporting: None

Program logic: This is a DEC 1070, TOPS–10 system resident routine. It provides the capabilities referred to in the Purpose section above. If the GRASP system is to be implemented on some other main frame, a comparable routine must be written or acquired. No listing is shown here.

SUBROUTINE NAME: DEFLST

Purpose: DEFLST outputs the category names to the user and allows him to indicate which categories are of interest.

Calling sequence: CALL DEFLST(&m,&n,CAT,NUMC,MC, LIST)

Arguments:

m—Statement (in calling routine) that will be branched to if no category numbers are given when asked for.

n—Statement (in calling routine) that will be branched to if an EOF is sensed in KEYBRD.

CAT—Contains the category names as read from unit 20 (the "definitions file").

NUMC—Number of categories selected by the user.

MC—Maximum length in words of a category name.

LIST—The category numbers selected.

Subroutines called: IFILE, KEYBRD, LENGTH, RLIST

Common data referenced: FNAMES, WHICH in /FILNAM/

Called by: NAME, DUMPIT

Error checking and reporting: All user response is checked for validity. If errors are detected, the response is requested again.

Program logic:

1. A call to IFILE associates the "definitions" file name with FORTRAN unit 20, and the category names are read from this file.
2. The user is asked if he is interested in all categories. His response is checked against "Y" or "N." If invalid, an error message is typed, and he is asked to respond again.
3. If the user's response was "Y," LIST is set to all the category numbers and control is returned to the calling routine.
4. If the response was "N," the user is asked to enter a list of category numbers of interest.
5. His response, contained in IMAGE, is passed to RLIST to generate the values of LIST.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE DEFLST(*,*,CAT,NUMC,MC,LIST)                          0000454
      COMMON /FILNAM/ FNAMES,WHICH,PAD                                 0000455
      INTEGER CAT(8,1 ),LIST(1),IMAGE(30),FNAMES(21),WHICH,PAD(4)      0000456
      DATA IYES,NO/'Y','N'/                                            0000457
      CALL IFILE(20,FNAMES(8+WHICH))                                   0000458
      READ (20) NCAT,MC,((CAT(I,J),I=1,MC),J=1,NCAT)                   0000459
   25 TYPE 30                                                          0000460
      CALL KEYBRD(&100,I,1)                                            0000461
      IF(I.EQ.IYES) GO TO 40                                           0000462
      IF(I.EQ.NO) GO TO 5                                              0000463
      TYPE 35                                                          0000464
      GO TO 25                                                         0000465
   40 NUMC=NCAT                                                        0000466
      DO 45 I=1,NUMC                                                   0000467
   45 LIST(I)=I                                                        0000468
      GO TO 85                                                         0000469
    5 TYPE 110                                                         0000470
      DO 10 J=1,NCAT                                                   0000471
      CALL LENGTH(CAT(1,J),MC,MCL)                                     0000472
   10 TYPE 120, J,(CAT(I,J),I=1,MCL)                                   0000473
      TYPE 130                                                         0000474
   20 CALL KEYBRD(&100,IMAGE,30)                                       0000475
      CALL RLIST(&20,IMAGE,LIST,NUMC,NCAT)                             0000476
      IF (NUMC.EQ.0) GO TO 90                                          0000477
   85 RETURN                                                           0000478
   90 RETURN 1                                                         0000479
  100 RETURN 2                                                         0000480
   30 FORMAT(' SHALL ALL CATEGORIES BE CONSIDERED? (YES OR NO): ',$)   0000481
   35 FORMAT(' YOUR REPLY WAS NOT UNDERSTOOD.')                        0000482
  110 FORMAT (' EACH RECORD HAS BEEN DIVIDED INTO THE FOLLOWING ','GENER0000483
     1AL CATEGORIES:'/8X,'CAT. #  CAT. NAME'/8X,'------  ---------')   0000484
  120 FORMAT (10X,I2,4X,9A5)                                           0000485
  130 FORMAT (' ENTER A LIST OF ASCENDING NUMBERS MATCHING ','YOUR CATEG0000486
     1ORIES OF INTEREST'/' (IE. 1,3,5 OR 2-5)')                        0000487
      END                                                              0000488
```

SUBROUTINE NAME: DUMPIT

Purpose: DUMPIT outputs to the terminal those values for all items present in a set of user-selected categories. The values are obtained from a user-selected file.

Calling sequence: CALL DUMPIT

Arguments: None

Subroutines called: OPREP, DEFLST, FINDGP, ACCESS, GETPUT, PAUSE, LENGTH, UNCODE, BINTYP, BLIST

Common data referenced:
NFILE in /IOUNIT/
NAMES, ITYPE in blank common

Called by: DRIVER

Error checking and reporting: None

Program logic:

1. Page size (NPAGE), input file name, and file unit are set up by a call to OPREP.
2. A call to DEFLST is made to determine categories to be dumped.
3. Calls to FINDGP are made to determine pointers (KLIST) for those items in the selected categories. As DEFLST and FINDGP used FORTRAN unit 20, the "definitions" file for the current data base, the unit is rewound.
4. A call to ACCESS is made to initialize character dictionary lookups.
5. Each record in the input file is obtained by GETPUT, the selected items are tested for nonblank characters, and their value is output.
6. The output algorithm is basically as follows:
   (a) Determine item type and switch to appropriate code section via a computed GOTO.
   (b) If type is integer, print under an I format.
   (c) If type is real, print under a G format.
   (d) If type is character, obtain string value by a call to ACCESS and print under an A format.
   (e) If type is multiple choice, obtain possible values by a call to BINTYP, and select the actual subset via a call to BLIST. Print this subset under an A format.
   (f) If type is qualified real, obtain value and qualifier via a call to UNCODE. Print under a G and A format.
   (g) After each line is printed, increment and test KOUNT against page size. If KOUNT is greater than page size, call PAUSE for a programmed pause and re-initialize KOUNT to zero.
   (h) After each record has been processed, print a line of asterisks as a record separator.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE DUMPIT                                                  0000489
      COMMON NAMES,ITYPE,IPTS,IPAD                                       0000490
      COMMON /IOUNIT/ NFILE,IOF                                          0000491
      DIMENSION ITYPE(500), BITEM(15,25), IREC(500), IQUAL(6),           0000492
     1          REC(500), TANK(25), LABEL(25), LIST(25)                 0000493
      DOUBLE PRECISION LABEL,NAMES1, NAMES(500)                          0000494
      INTEGER TANK,CAT(8,17),KLIST(17),SELECT(500),IPTS(500)             0000495
      LOGICAL PNTHDG,NEWCAT,HIT                                          0000496
      EQUIVALENCE (REC,IREC),(IVAL,VAL),(TANK,LIST),(IBLNK,IQUAL)        0000497
      DATA IQUAL/' ','G','H','L','N','T'/                                0000498
      KOUNT=0                                                            0000499
      CALL OPREP(&200,&205,NPAGE)                                        0000500
      CALL DEFLST(&200,&200,CAT,NUMC,MC,KLIST)                           0000501
      NTOT=0                                                             0000502
      DO 20 K=1,NUMC                                                     0000503
      CALL FINDGP(&200,KLIST(K),I,J,NG,IREC)                             0000504
      READ (20)                                                          0000505
      DO 10 I=1,NG                                                       0000506
      NTOT=NTOT+1                                                        0000507
   10 SELECT(NTOT)=2*IREC(I)                                             0000508
      SELECT(NTOT-NG+1)=SELECT(NTOT-NG+1)+1                              0000509
   20 CONTINUE                                                           0000510
      REWIND 20                                                          0000511
      CALL ACCESS(II,IVAL,TANK,J,1)                                      0000512
   40 CALL GETPUT(&200,IREC,1)                                           0000513
      HIT=.FALSE.                                                        0000514
      KNT=0                                                              0000515
      DO 190 JJ=1,NTOT                                                   0000516
      II=SELECT(JJ)/2                                                    0000517
      NEWCAT=SELECT(JJ).NE.2*II                                          0000518
      IF (NEWCAT) KNT=KNT+1                                              0000519
      PNTHDG=PNTHDG.OR.NEWCAT                                            0000520
```

```
      NAMESI=NAMES(II)                                                  0000521
      IVAL=IREC(II)                                                     0000522
      IF (IVAL.EQ.IBLNK) GO TO 190                                      0000523
      HIT=.TRUE.                                                        0000524
      KOUNT=KOUNT+1                                                     0000525
      IF (KOUNT.LE.NPAGE) GO TO 50                                      0000526
      KOUNT=0                                                           0000527
      CALL PAUSE(&200)                                                  0000528
   50 KIND=ITYPE(II)                                                    0000529
      IF(.NOT.PNTHDG) GO TO 55                                          0000530
      KL=KLIST(KNT)                                                     0000531
      CALL LENGTH(CAT(1,KL),MC,MCL)                                     0000532
      TYPE 210,(CAT(I,KL),I=1,MCL)                                      0000533
      PNTHDG=.FALSE.                                                    0000534
55      GO TO (60,70,80,170,160),KIND                                   0000535
   60 TYPE 230, NAMESI,IVAL                                             0000536
      GO TO 190                                                         0000537
   70 TYPE 240, NAMESI,VAL                                              0000538
      GO TO 190                                                         0000539
   80 CALL ACCESS(II,IVAL,TANK,J,2)                                     0000540
      TYPE 250, NAMESI,(TANK(I),I=1,J)                                  0000541
      GO TO 190                                                         0000542
  160 VAL=UNCODE(VAL,IQ)                                                0000543
      TYPE 240,NAMESI,VAL,IQUAL(IQ)                                     0000544
      GO TO 190                                                         0000545
  170 CALL BINTYP(II,LABEL,BITEM,K,M)                                   0000546
      KOUNT=KOUNT+1                                                     0000547
      TYPE 250, NAMESI                                                  0000548
      CALL BLIST(LIST,NUMS,IVAL)                                        0000549
      DO 180 I=1,NUMS                                                   0000550
      J=LIST(I)                                                         0000551
  180 TYPE 260, LABEL(J),(BITEM(L,J),L=1,K)                             0000552
  190 CONTINUE                                                          0000553
      IF (HIT) TYPE 220                                                 0000554
      GO TO 40                                                          0000555
  200 REWIND NFILE                                                      0000556
      REWIND 20                                                         0000557
205     RETURN                                                          0000558
  210 FORMAT (' CATEGORY: ',8A5)                                        0000559
  220 FORMAT (1X,3(8H********))                                         0000560
  230 FORMAT (2X,A8,1X,I9)                                              0000561
  240 FORMAT (2X,A8,1X,1PG12.5,A1)                                      0000562
  250 FORMAT (2X,A8,1X,12A5/11X,12A5)                                   0000563
  260 FORMAT (5X,A8,15A4)                                               0000564
      END                                                               0000565
```

FUNCTION NAME: EVAL

Purpose: By using a particular set of values as operands, EVAL evaluates a previously parsed Reverse-Polish-form arithmetic expression.

Calling sequence: VAL=EVAL(VALUES,TYPE,POLISH, I,BLNK)

Arguments:

VALUES—Set of operand values.

TYPE, POLISH—Arrays containing the encoded Reverse-Polish form of the expression to be evaluated. The encoding is as follows: Let ITY be the I'th element of TYPE. If ITY=0, the I'th element of POLISH is a numeric constant. If ITY>0, ITY is an index to the array VALUES. If ITY<0, ITY corresponds to an arithmetic operator or function.

I—Gives the length of the arrays TYPE and POLISH.

BLNK—Logical variable set to TRUE if any operand with a blank value is sensed.

Subroutines called: UNCODE

Common data referenced: ITYPE in blank common

Called by: COLPNT, ROWPNT

Error checking and reporting:

1. Division by zero attempted.
2. Log of a nonpositive value attempted.
3. Square root of a negative value attempted.

Program logic: A push-down stack technique is used to evaluate the Reverse-Polish string contained in TYPE and POLISH. TYPE is scanned, an element at a time, pushing operand values down on the stack until an operator is sensed. Either the top or topmost two stack elements are then used as operands resulting in a new topmost-stack element which is the resulting value of the operator. Unary operators/functions (absolute value, ABS; square root, SQRT; logarithm, LOG; square, SQR; ten exponent, TEN: minus, −) operate on the top stack element. Binary operators (+, −, *, /) operate on the topmost-two stack elements. After all elements of TYPE have been processed, the stack should have one value in it. This value, the result, is returned. If a blank operand value is detected, the flag BLNK is turned on and zero is returned.

G R A S P S O U R C E P R O G R A M

```
      FUNCTION EVAL(VALUES,TYPE,POLISH,I,BLNK)                          0000566
      COMMON NAMES,ITYPE,IPTS,IPAD                                      0000567
      DIMENSION NAMES(500), VALUES(1), POLISH(1), STACK(41), TYPE(1), IT0000568
     1YPE(500),IPTS(500)                                                0000569
      DOUBLE PRECISION NAMES                                            0000570
      INTEGER TOP,TYPE,VALUES                                           0000571
      LOGICAL BLNK                                                      0000572
      EQUIVALENCE (VAL,IVAL)                                            0000573
      DATA IBLNK/' '/                                                   0000574
      BLNK=.FALSE.                                                      0000575
      TOP=0                                                             0000576
      DO 190 J=1,I                                                      0000577
      INDEX=TYPE(J)                                                     0000578
      IF (INDEX) 40,30,10                                               0000579
   10 TOP=TOP+1                                                         0000580
      IVAL=VALUES(INDEX)                                                0000581
      IF (IVAL.EQ.IBLNK) GO TO 195                                      0000582
      IF (ITYPE(INDEX).EQ.2) GO TO 15                                   0000583
      IF(ITYPE(INDEX).NE.5) GO TO 20                                    0000584
      VAL=UNCODE(VAL,IQ)                                                0000585
   15 STACK(TOP)=VAL                                                    0000586
      GO TO 190                                                         0000587
   20 STACK(TOP)=IVAL                                                   0000588
      GO TO 190                                                         0000589
   30 TOP=TOP+1                                                         000059C
      STACK(TOP)=POLISH(J)                                              0000591
      GO TO 190                                                         0000592
   40 IF (INDEX.GE.-4) GO TO 130                                        0000593
      INDEX=INDEX+11                                                    0000594
      GO TO (50,60,80,100,110,120), INDEX                               0000595
   50 STACK(TOP)=ABS(STACK(TOP))                                        0000596
      GO TO 190                                                         0000597
```

```
 60 TV=STACK(TOP)                                                        0000598
    IF (TV.GE.0.) GO TO 70                                               0000599
    TYPE 230                                                             0000600
    GO TO 200                                                            0000601
 70 STACK(TOP)=SQRT(TV)                                                  0000602
    GO TO 190                                                            0000603
 80 TV=STACK(TOP)                                                        0000604
    IF (TV.GT.0.) GO TO 90                                               0000605
    TYPE 220                                                             0000606
    GO TO 200                                                            0000607
 90 STACK(TOP)=ALOG10(TV)                                                0000608
    GO TO 190                                                            0000609
100 STACK(TOP)=STACK(TOP)**2                                             0000610
    GO TO 190                                                            0000611
110 STACK(TOP)=10.**STACK(TOP)                                           0000612
    GO TO 190                                                            0000613
120 STACK(TOP)=-STACK(TOP)                                               0000614
    GO TO 190                                                            0000615
130 INDEX=INDEX+5                                                        0000616
    VT=STACK(TOP)                                                        0000617
    TOP=TOP-1                                                            0000618
    GO TO (150,140,180,170), INDEX                                       0000619
140 STACK(TOP)=STACK(TOP)*VT                                             0000620
    GO TO 190                                                            0000621
150 IF (VT.NE.0.0) GO TO 160                                             0000622
    TYPE 240                                                             0000623
    GO TO 200                                                            0000624
160 STACK(TOP)=STACK(TOP)/VT                                             0000625
    GO TO 190                                                            0000626
170 STACK(TOP)=STACK(TOP)+VT                                             0000627
    GO TO 190                                                            0000628
180 STACK(TOP)=STACK(TOP)-VT                                             0000629
190 CONTINUE                                                             0000630
    IF (TOP .NE. 1) GO TO 200                                            0000631
    EVAL=STACK(1)                                                        0000632
    GO TO 210                                                            0000633
195 BLNK=.TRUE.                                                          0000634
200 EVAL=0.0                                                             0000635
210 RETURN                                                               0000636
220 FORMAT (' ATTEMPTED TO TAKE LOG OF A ZERO OR NEG. VALUE.')           0000637
230 FORMAT (' ATTEMPTED TO TAKE SQRT OF A NEGATIVE VALUE.')              0000638
240 FORMAT (' DIVIDE BY ZERO ATTEMPTED.')                                0000639
    END                                                                  0000640
```

SUBROUTINE NAME: FDRIVE

Purpose: FDRIVE provides for the single-pass computation of all implemented mathematical or statistical functions. Using this routine to make all calls to the subroutines corresponding to implemented functions simplifies the addition of new functions.

Calling sequence: CALL FDRIVE(ISWTCH)

Arguments:

ISWTCH—Integer code passed to called subroutines indicating phase 1, 2, or 3. The phases are initialization, body, and postprocessing.

Subroutines called: MEAN, FIT

Common data referenced: None

Called by: FTNC

Error checking and reporting: None

Program logic: This routine merely makes calls to the functions selected by the user via computed GOTO.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE FDRIVE(ISWTCH)                                        0000641
      COMMON /FTNCOM/ TAGS,IREC,ARGS,NARGS,IFTN,NFTN                   0000642
      DOUBLE PRECISION TAGS(5,5)                                       0000643
      INTEGER ARGS(6,5),NARGS(5),IFTN(5),IREC(500)                     0000644
      DO 30 J=1,NFTN                                                   0000645
      I=IFTN(J)                                                        0000646
      GO TO (10,20,30,30,30,30), I                                     0000647
   10 CALL MEAN(J,ISWTCH)                                              0000648
      GO TO 30                                                         0000649
   20 CALL FIT(J,ISWTCH)                                               0000650
   30 CONTINUE                                                         0000651
      RETURN                                                           0000652
      END                                                              0000653
```

SUBROUTINE NAME: FILES

Purpose: FILES prompts for and accepts a data-base name. After the name has been provided and verified, the Mask file of data-base characteristics associated with that name is read. Some preliminary processing is done on these characteristics.

Calling sequence: CALL FILES(&n)

Argument:

n—Statement (in caller) to which a branch is made if the nonstandard return from KEYBRD is taken (namely, an EOF is sensed).

Subroutines called: KEYBRD, IFILE, BINIT, DEFINE, PACK

Common data referenced:

NAMES, ITYPE, IPTS, IDIM in blank common.
MASTER, MASK, DFILE, BFILE, NUMF, NUMI, IDIMS in /FILNAM/.
I, IDPT in /DACOMM/.

Called by: DRIVER

Error checking and reporting: The data-base name entered by the user is compared with the list (MASTER) of those available. If the data-base name is not recognized, an error message is typed.

Program Logic:

1. A data-base name is prompted for and accepted via KEYBRD, and is then packed into FILEID and compared with the list (MASTER) of available names. If not found, an error message and the list of available names is typed followed by a prompt for another name.
2. Once the data base has been established, its corresponding Mask file is read to fill the blank common area.
3. The item names from MASK are sorted via a call to BINIT and the array of pointers (IPTS) to the sorted NAMES is returned from BINIT.
4. NAMES is restored to its unsorted form and DEFINE is called to associate the name of the direct-access character dictionary with unit 21. IFILE is then called to associate the name of the multiple-choice file with unit 22.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE FILES(*)                                                0000654
      COMMON NAMES,ITYPE,IPTS,IDIM                                       0000655
      COMMON /FILNAM/ MASTER,MASK,DEFTN,DFILE,BFILE,NUMF,NUMI,IDIMS      0000656
      COMMON /DACOMM/ I,IDPT                                             0000657
      DIMENSION ITYPE(500), IPTS(500), IDPT(500), IDIMS(4)               0000658
      DOUBLE PRECISION NAMES(500),VNAMES(500)                            0000659
      INTEGER MASTER(4),MASK(4),DEFTN(4),DFILE(4),BFILE(4),FILEID        0000660
      DATA I20,I21,I22/20,21,22/                                         0000661
    1 TYPE 11                                                            0000662
      CALL KEYBRD(&999,ITYPE,5)                                          0000663
      CALL PACK(ITYPE,FILEID,5,5)                                        0000664
      DO 3 J=1,NUMF                                                      0000665
      IF(MASTER(J).EQ.FILEID) GO TO 5                                    0000666
    3 CONTINUE                                                           0000667
      TYPE 4,FILEID,(MASTER(I),I=1,NUMF)                                 0000668
      GO TO 1                                                            0000669
    5 NUMI=J                                                             0000670
      CALL IFILE(I20,MASK(NUMI))                                         0000671
      IDIM=IDIMS(NUMI)                                                   0000672
      READ(I20) (ITYPE(I),I=1,IDIM),(IDPT(I),I=1,IDIM),                  0000673
     1 (NAMES(I),I=1,IDIM)                                               0000674
      REWIND I20                                                         0000675
      DO 10 I=1,IDIM                                                     0000676
   10 VNAMES(I)=NAMES(I)                                                 0000677
      CALL BINIT(NAMES,IPTS,IDIM)                                        0000678
      DO 20 I=1,IDIM                                                     0000679
   20 NAMES(I)=VNAMES(I)                                                 0000680
      CALL DEFINE(I21,27,I,DFILE(NUMI),"412,"176)                        0000681
      CALL IFILE(I22,BFILE(NUMI))                                        0000682
      RETURN                                                             0000683
  999 RETURN 1                                                           0000684
    4 FORMAT(1X,A6,'NOT AN AVAILABLE DATA BASE NAME. ',                  0000685
     1 'USE ONE OF THE FOLLOWING:'/(1X,A5))                              0000686
   11 FORMAT(' ENTER DATA BASE NAME: ',$)                                0000687
      END                                                                0000688
```

SUBROUTINE NAME: FIND

Purpose: FIND is used to look up a word in a "hash coded" table and return a code associated with its position. The "Linear quotient" technique, as described by Bell and Kaman (1970), is used.

Calling sequence: CALL FIND (&n,ISYMBL,KODE,CHARS, CODES,M)

Arguments:

n—Statement (in caller) to which a branch is taken if ISYMBL is not in CHARS.

ISYMBL—Word to be looked up.

KODE—Integer from the position in CODES corresponding to the position of ISYMBL in CHARS.

CHARS—"Hash coded" table of words to be looked up.

CODES—Table corresponding to CHARS giving the original position of the word to be looked up.

M—Table size (a prime number) for CHARS and CODES.

Subroutines called: None

Common data referenced: None

Called by: LOGEXP, PARSE

Error checking and reporting: If ISYMBL is not in CHARS, the nonstandard return is taken.

Program logic: An initial location I and displacement J is determined from the internal machine representation of the word stored in ISYMBL. Initial and successive probes to CHARS are made until an empty location is probed. If ISYMBL is not in CHARS, the nonstandard return is taken. If ISYMBL is found in CHARS, the corresponding element of CODES is returned in KODE.

GRASP SOURCE PROGRAM

```
      SUBROUTINE FIND(*,ISYMBL,KODE,CHARS,CODES,M)                    0000689
      INTEGER CHARS(1),CODES(1),FILLER                                0000690
      DATA FILLER/'VOID'/                                             0000691
      L=IABS(ISYMBL)                                                  0000692
      J=L/M                                                           0000693
      I=L-M*J                                                         0000694
      IF (MOD(J,M).EQ.0) J=1                                          0000695
   50 ICHAR=CHARS(I+1)                                                0000696
      IF (ICHAR.EQ.ISYMBL) GO TO 60                                   0000697
      IF (ICHAR.EQ.FILLER) RETURN 1                                   0000698
      I=MOD(I+J,M)                                                    0000699
      GO TO 50                                                        0000700
   60 KODE=CODES(I+1)                                                 0000701
      RETURN                                                          0000702
      END                                                             0000703
```

SUBROUTINE NAME: FINDGP

Purpose: FINDGP positions the file associated with FORTRAN unit 20 to a particular record paid associated with a category. Data concerning that category are returned to the caller.

Calling sequence: CALL FINDGP(&n,KNUM,NUM,MAXL, NG,GROUP)

Arguments:

n—Statement number (in caller) to which control is passed if an EOF is sensed on unit 20.

KNUM—The category number to which the file will be be positioned.

NUM—The number of descriptions in this category.

MAXL—The maximum length (in words) for a description.

NG—Number of items referred to in this category.

GROUP—List of item pointers associated with this category.

Subroutines called: IFILE

Common data referenced: FNAMES, WHICH in/FILNAM/

Called by: NAME, DUMPIT

Error checking and reporting: None

Program logic:

1. The next record on unit 20 is read, giving a category number KK and values for the last four arguments.
2. KK is tested against KNUM.
   (a) If KK<KNUM, record pairs are skipped up to the one of interest.
   (b) If KK=KNUM, return.
   (c) If KK>KNUM, rewind unit 20 and reassociate it with the correct name via a call to IFILE having FNAMES and WHICH as arguments. Next, reposition the file to the number pair of interest.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE FINDGP(*,KNUM,NUM,MAXL,NG,GROUP)                        0000704
      COMMON /FILNAM/ FNAMES,WHICH,PAD                                   0000705
      INTEGER GROUP(1),FNAMES(21),WHICH,PAD(4)                           0000706
      DATA NDEF/20/                                                      0000707
30    READ (NDEF,END=90) KK,NUM,MAXL,NG,(GROUP(I),I=1,NG)                0000708
      IF (KK-KNUM) 40,80,70                                              0000709
40    J=KNUM-KK-1                                                        0000710
50    READ(NDEF)                                                         0000711
      IF (J.LT.1) GO TO 30                                               0000712
      DO 60 I=1,J                                                        0000713
      READ(NDEF)                                                         0000714
60    READ(NDEF)                                                         0000715
      GO TO 30                                                           0000716
70    REWIND NDEF                                                        0000717
      CALL IFILE(NDEF,FNAMES(8+WHICH))                                   0000718
      J=KNUM                                                             0000719
      GO TO 50                                                           0000720
80    RETURN                                                             0000721
90    RETURN 1                                                           0000722
      END                                                                0000723
```

SUBROUTINE NAME: FIT

Purpose: FIT is used to provide a least-square linear fit between two items within a selected file.

Calling sequence: CALL FIT(J,ISWTCH)

Arguments:

J—Pointer used to retrieve argument values from the common area /FTNCOM/.

ISWTCH—Switch indicating which of three parts (initialization, body, postprocessing) of the code is to be executed.

Subroutines called: UNCODE

Common data referenced:

ITYPE in blank common

TAGS, IREC, ARGS, NARGS, in /FTNCOM/

Called by: FDRIVE

Error checking and reporting: If two arguments are not given, an appropriate error message is typed and return is immediate. If the computation would yield an infinite slope, that message is typed.

Program logic: The value of ISWTCH determines which of three sections of the code is executed. If ISWTCH=1, the number of arguments is checked, and various sums are set to zero. If ISWTCH=2, the error flag is tested. If not set, the appropriate values of the arguments are tested for nonblank. If nonblank, they are added to the appropriate sums. If ISWTCH=3, the slope, intercept, and correlation coefficient are calculated (if possible), using the sums previously determined. They are then printed out using the appropriate item names. All summations and least-square determinations are done using double-precision arithmetic to minimize the round-off effects introduced by performing the computation using only one pass on the data.

Assuming the function FIT X,Y had been issued, the calculations are performed using the following formula:

$$D=N\cdot\Sigma X^2-\Sigma X\cdot\Sigma X$$

$$B_1=(N\cdot\Sigma XY-\Sigma X\cdot\Sigma Y)/D$$

$$B_0=(\Sigma Y-B_1\cdot\Sigma X)/N$$

$$C=D\cdot B_1/\sqrt{D\cdot(N\cdot\Sigma Y^2-\Sigma Y\cdot\Sigma Y)}$$

where:

B_0=intercept,

B_1=slope,

C=correlation coefficient, and

N=number of nonblank X, Y points.

GRASP SOURCE PROGRAM

```
      SUBROUTINE FIT(J,ISWTCH)                                          0000724
      COMMON NAMES,ITYPE,IPTS,IPAD                                      0000725
      COMMON /FTNCOM/ TAGS,IREC,ARGS,NARGS,IFTN,NFTN                    0000726
      DOUBLE PRECISION NAMES(500),TAGS(5,5),SUMX,SUMY,SUMXY,            0000727
     1 SUMXS,SUMYS,D,FN,V1,V2                                           0000728
      INTEGER IPTS(500)                                                 0000729
      INTEGER ARGS(6,5),NARGS(5),IFTN(5),ITYPE(500),IREC(500)           0000730
      LOGICAL ERR                                                       0000731
      EQUIVALENCE (IVAL1,VAL1), (IVAL2,VAL2)                            0000732
      DATA IBLNK/' '/                                                   0000733
      IF(ISWTCH-2) 2,4,6                                                0000734
    2 ERR=NARGS(J).NE.2                                                 0000735
      IF (ERR) GO TO 20                                                 0000736
      N=0                                                               0000737
      SUMX=0.0D0                                                        0000738
      SUMY=0.0D0                                                        0000739
      SUMXY=0.0D0                                                       0000740
      SUMXS=0.0D0                                                       0000741
      SUMYS=0.0D0                                                       0000742
      GO TO 30                                                          0000743
    4 IF (ERR) GO TO 30                                                 0000744
      IVAL1=IREC(ARGS(2,J))                                             0000745
      IF (IVAL1.EQ.IBLNK) GO TO 30                                      0000746
      IVAL2=IREC(ARGS(3,J))                                             0000747
      IF (IVAL2.EQ.IBLNK) GO TO 30                                      0000748
      V1=IVAL1                                                          0000749
      V2=IVAL2                                                          0000750
      IF (ITYPE(ARGS(2,J)).EQ.2) V1=VAL1                                0000751
      IF (ITYPE(ARGS(3,J)).EQ.2) V2=VAL2                                0000752
      IF(ITYPE(ARGS(2,J)).EQ.5) V1=UNCODE(VAL1,IQ)                      0000753
      IF(ITYPE(ARGS(3,J)).EQ.5) V2=UNCODE(VAL2,IQ)                      0000754
      N=N+1                                                             0000755
      SUMX=SUMX+V1                                                      0000756
```

```
      SUMXS=SUMXS+V1*V1                                                    0000757
      SUMXY=SUMXY+V1*V2                                                    0000758
      SUMY=SUMY+V2                                                         0000759
      SUMYS=SUMYS+V2*V2                                                    0000760
      GO TO 30                                                             0000761
    6 IF (ERR) GO TO 30                                                    0000762
      FN=N                                                                 0000763
      D=FN*SUMXS-SUMX*SUMX                                                 0000764
      IF (D.EQ.0.0D0) GO TO 10                                             0000765
      B1=(FN*SUMXY-SUMX*SUMY)/D                                            0000766
      B0=(SUMY-B1*SUMX)/FN                                                 0000767
      CC=(FN*SUMXY-SUMX*SUMY)/DSQRT((FN*SUMXS-SUMX*SUMX)*(FN*SUMYS-SUMY*0000768
     1SUMY))                                                               0000769
      TYPE 40, N,TAGS(1,J),TAGS(2,J),B1,B0,CC                              0000770
      GO TO 30                                                             0000771
   10 TYPE 50, N                                                           0000772
      GO TO 30                                                             0000773
   20 TYPE 60                                                              0000774
   30 RETURN                                                               0000775
   40 FORMAT (/1X,I5,' POINTS USED TO FIT ',A8,'TO ',A8/' SLOPE=',1PE12.0000776
     15,' INTERCEPT=',E12.5,' CORR. COEFF.=',E12.5)                        0000777
   50 FORMAT (' UNABLE TO CALCULATE FIT WITH',I5,' POINTS')                0000778
   60 FORMAT (' THE FIT FUNCTION MUST HAVE 2 ARGUMENTS')                   0000779
      END                                                                  0000780
```

SUBROUTINE NAME: FTNC

Purpose: This routine acts as a driver for the processing of the FUNCTION command. It accepts (via KEYBRD) the function names and arguments, sets up system-required input-file information, and is then used to supply input records to the routines that actually calculate the requested functions.

Calling sequence: CALL FTNC(&n)

Argument:

n—Statement (in caller) to which a branch is made if the nonstandard return (EOF) from KEYBRD is taken.

Subroutines called: KEYBRD, OBEY, DECOMP, FDRIVE, GETPUT, PACK

Common data referenced:

FNAMES, WHICH in /FILNAM/
TAGS, IREC, ARGS, NARGS, IFTN, NFTN in /FTNCOM/
NFILE in /IOUNIT/

Called by: DRIVER

Error checking and reporting: Function names entered by the user are checked against a list of those available. If an invalid function name is entered, that message is typed.

Program logic:

1. A prompted input-file name is obtained via KEYBRD, packed via a call to PACK, and associated with the FORTRAN input unit number via OBEY.
2. A list of the implemented function names is typed along with a request to enter the names of desired functions and their corresponding arguments.
3. As each function name is entered (via KEYBRD), it and its argument names are identified via DECOMP.
4. After the names have been entered and identified, a call to FDRIVE using an argument of 1 is issued to perform initialization.
5. Each record of the selected file is obtained via GETPUT and processed via a call to FDRIVE using an argument of 2.
6. Finally, a call to FDRIVE using an argument of 3 is made to accomplish any wrap-up processing associated with the selected function, and the input file is rewound.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE FTNC(*)                                                0000781
      COMMON /FILNAM/ FNAMES,WHICH,PAD                                  0000782
      COMMON /FTNCOM/ TAGS,IREC,ARGS,NARGS,IFTN,NFTN                    0000783
      COMMON /IOUNIT/ NFILE,IPAD                                        0000784
      DOUBLE PRECISION TAGS(5,5)                                        0000785
      INTEGER DFAULT,FILE,DBLNK,FNAMES(21),WHICH,PAD(4),IFTN(5),        0000786
     1EQUATE(4),FTNS(5),ARGS(6,5),IMAGE(80),NARGS(5),IREC(500),         0000787
     2 PROMPT(5)                                                        0000788
      LOGICAL ANY,GOOD                                                  0000789
      DATA DBLNK,EQUATE/' ','EQUA','TE 1','1',' '/                      0000790
      DATA IMPLTD,FTNS/2,'MEAN','FIT',3*' '/                            0000791
      DATA PROMPT/'1.','2.','3.','4.','5.'/                             0000792
      EQUIVALENCE (EQUATE(4),FILE)                                      0000793
      TYPE 100                                                          0000794
      DFAULT=FNAMES(WHICH)                                              0000795
      CALL KEYBRD(&90,IMAGE,5)                                          0000796
      CALL PACK(IMAGE,FILE,5,5)                                         0000797
      IF (FILE.EQ.DBLNK) FILE=DFAULT                                    0000798
      CALL OBEY(&85,EQUATE,4)                                           0000799
      TYPE 120, (FTNS(I),I=1,IMPLTD)                                    0000800
      TYPE 130                                                          0000801
   10 NFTN=1                                                            0000802
   20 TYPE 150, PROMPT(NFTN)                                            0000803
      CALL KEYBRD(&90,IMAGE,80)                                         0000804
      CALL DECOMP(&20,&30,IMAGE,TAGS(1,NFTN),ARGS(1,NFTN),NARG)         0000805
      NARGS(NFTN)=NARG-1                                                0000806
      NFTN=NFTN+1                                                       0000807
      IF (NFTN-IMPLTD-1) 20,30,30                                       0000808
   30 NFTN=NFTN-1                                                       0000809
      IF (NFTN.EQ.0) RETURN                                             0000810
      ANY=.FALSE.                                                       0000811
      DO 60 I=1,NFTN                                                    0000812
      GOOD=.TRUE.                                                       0000813
      INAME=ARGS(1,I)                                                   0000814
      DO 40 J=1,IMPLTD                                                  0000815
      IF (INAME.EQ.FTNS(J)) GO TO 50                                    0000816
```

```
   40 CONTINUE                                                                  0000817
      TYPE 160, INAME                                                           0000818
      J=6                                                                       0000819
      GOOD=.FALSE.                                                              0000820
   50 IFTN(I)=J                                                                 0000821
   60 ANY=ANY.OR.GOOD                                                           0000822
      IF (.NOT.ANY) GO TO 10                                                    0000823
      CALL FDRIVE(1)                                                            0000824
   70 CALL GETPUT(&80,IREC,1)                                                   0000825
      CALL FDRIVE(2)                                                            0000826
      GO TO 70                                                                  0000827
   80 CALL FDRIVE(3)                                                            0000828
      REWIND NFILE                                                              0000829
85       RETURN                                                                0000830
   90 RETURN 1                                                                  0000831
  100 FORMAT (' ENTER NAME OF FILE: ',$)                                        0000832
  120 FORMAT (' FUNCTIONS AVAILABLE AT THIS TIME ARE:'/1X,5A8)                  0000833
  130 FORMAT (' ENTER FUNCTION NAMES AND CORRESPONDING ARGUMENTS. ')            0000834
  150 FORMAT (1X,A3,$)                                                          0000835
  160 FORMAT (1X,A5,' IS NOT AN AVAILABLE FUNCTION AND HAS ','BEEN IGNOR0000836
     1ED.')                                                                     0000837
      END                                                                       0000838
```

SUBROUTINE NAME: GETPUT

Purpose: GETPUT is used to read and unpack a record from the current input file, or to write the last-packed record obtained from the current input file on the current output file.

Calling sequence: CALL GETPUT(&n,IREC,ISWTCH)

Arguments:

n—Statement (in caller) to which a branch is made if an EOF is sensed on the current input file.

IREC—Contains the expanded record from the current input file.

ISWTCH—Switch indicating whether record is to be read or written.

Subroutines called: None

Common data referenced: IN, IOUT in /IOUNIT/

Called by: COLPNT, DUMPIT, FTNC, RETRVE, ROWPNT

Error checking and reporting: None

Program logic:

1. If ISWTCH=1, the next input record is read into TANK on the unit referenced by IN in /IOUNIT/. If an EOF is sensed, the nonstandard return is taken. If ISWTCH>1, the last-read input record is output on the unit referenced by IOUT in /IOUNIT/.
2. The first word of TANK is assumed to be of type INTEGER and corresponds to the first word of IREC. The last 2 bits contain the type of the next word in TANK. TYPE values are:

TYPE	*Indicates*
1	Next word is an integer value.
2	Next word is a real value.
3	Next word is a blank counter.

   The value part of types 1 and 3 is in the leading bits (that is, 2-bit truncation).
3. If the word is of type REAL, it may be visualized as being composed of an integer and a fraction. The type for the next word is in the last 2 bits of the integer part and the associated real value is obtained by adding the fractional part to the leading bits of the integer part.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE GETPUT(*,IREC,ISWTCH)                                  0000839
      COMMON /IOUNIT/ IN,IOUT                                           0000840
      INTEGER IREC( 1 ),ITANK(150),BLANK,TYPE                           0000841
      REAL TANK(150)                                                    0000842
      EQUIVALENCE (FRACT,IVAL), (ITANK(1),TANK(1))                      0000843
      DATA BLANK/' '/                                                   0000844
      IF(ISWTCH-2) 5,70,70                                              0000845
    5 READ (IN,END=110) NUM,(TANK(I),I=1,NUM)                           0000846
      TYPE=1                                                            0000847
      IPT=1                                                             0000848
      DO 50 I=1,NUM                                                     0000849
      IF(TYPE.LT.0) TYPE=-TYPE                                          0000850
      IF (TYPE-2) 10,20,30                                              0000851
   10 IVAL=ITANK(I)/4                                                   0000852
      TYPE=ITANK(I)-4*IVAL                                              0000853
      GO TO 25                                                          0000854
   20 TYPE=TANK(I)                                                      0000855
      FRACT=TANK(I)-TYPE                                                0000856
      J=TYPE/4                                                          0000857
      TYPE=TYPE-4*J                                                     0000858
      FRACT=J+FRACT                                                     0000859
   25 IREC(IPT)=IVAL                                                    0000860
      IPT=IPT+1                                                         0000861
      GO TO 50                                                          0000862
   30 J=ITANK(I)/4                                                      0000863
      TYPE=ITANK(I)-4*J                                                 0000864
      DO 40 K=1,J                                                       0000865
      IREC(IPT)=BLANK                                                   0000866
   40 IPT=IPT+1                                                         0000867
   50 CONTINUE                                                          0000868
   60 RETURN                                                            0000869
   70 WRITE(IOUT)NUM,(TANK(I),I=1,NUM)                                  0000870
      GO TO 60                                                          0000871
  110 RETURN 1                                                          0000872
      END                                                               0000873
```

SUBROUTINE NAME: HELP

Purpose: HELP types the names and descriptions of GRASP commands available to the user.

Calling sequence: CALL HELP(WORDS)

Argument:

WORDS—An array containing the command names that may be issued by the user.

Subroutines called: None

Common data referenced: None

Called by: DRIVER

Error checking and reporting: None

Program logic: The text associated with each of the commands is initialized in DATA statements. Each command name (in WORDS) is typed with its corresponding description in TEXT.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE HELP(WORDS)                                                    0000874
      INTEGER WORDS(1)                                                          0000875
      DOUBLE PRECISION TEXT(11,11),MSG1(11),MSG2(11),MSG3(11),                  0000876
     1 MSG4(11),MSG5(11),MSG6(11),MSG7(11),MSG8(11),MSG9(11),                   0000877
     2 MSG10(11),MSG11(11)                                                      0000878
      EQUIVALENCE (MSG1(1),TEXT(1,1)),(MSG2(1),TEXT(1,2)),                      0000879
     1 (MSG3(1),TEXT(1,3)),(MSG4(1),TEXT(1,4)),(MSG5(1),TEXT(1,5)),             0000880
     2 (MSG6(1),TEXT(1,6)),(MSG7(1),TEXT(1,7)),(MSG8(1),TEXT(1,8)),             0000881
     3 (MSG9(1),TEXT(1,9)),(MSG10(1),TEXT(1,10)),(MSG11(1),TEXT(1,11))         0000882
      DATA MSG1/'- INITIA','TES THE','REQUEST','FOR RETR','IEVAL CR',           0000883
     1'ITERIA T','O BE ENT','ERED IN','THE FORM',': NAME R','EL VALUE'/         0000884
      DATA MSG2/'- INITIA','TES THE','REQUEST','FOR A LO','GICAL EX',           0000885
     1'PRESSION',' TO BE E','NTERED','USING LO','GICAL OP','ERATORS.'/          0000886
      DATA MSG3/'- INITIA','TES THE','SEARCH O','F A FILE',' BASED U',          0000887
     1'PON PREV','IOUSLY E','NTERED','CONDITIO','NS AND L','OGIC. '/            0000888
      DATA MSG4/'- ALLOWS',' THE USE','R TO LIS','T SELECT','ED VALUE',         0000889
     1'S (VARIA','BLE NAME','S WILL','BE ASKED',' FOR) IN',' A FILE.'/          0000890
      DATA MSG5/'- ALLOWS',' THE USE','R TO SEL','ECT OR C','HANGE TH',         0000891
     1 'E DATA B','ASE TO B','E USED.',3*'      '/                              0000892
      DATA MSG6/'- TERMIN','ATES THE',' SYSTEM.',' ENTERIN','G   IN R',         0000893
     1'ESPONSE','TO A PRO','MPT WILL','ALSO STO','P THE SY','STEM. '/           0000894
      DATA MSG7/'- USED T','O PRINT','ITEM NAM','ES, THEI','R TYPES',           0000895
     1'AND DEFI','NITIONS','IN A  ','SELECTED',' SET OF','GROUPS.'/             0000896
      DATA MSG8/'- USED T','O OBTAIN',' THE ABO','VE COMMA','ND DEFIN',         0000897
     1'ITIONS.',5*'      '/                                                     0000898
      DATA MSG9/'- LISTS','THE FILE','S WHICH','HAVE BEE','N USED A',           0000899
     1'S WELL A','S THE CO','NDITIONS','AND LOGI','C ENTERE','D.    '/          0000900
      DATA MSG10/'- PRINTS',' ALL ITE','MS PRESE','NT FOR E','ACH RECO',        0000901
     1'RD IN A','SELECTED',' FILE.','WAITS AF','TER EACH',' N LINES'/           0000902
      DATA MSG11/'- PROVID','ES FOR T','HE COMPU','TATION O','F FUNCTI',        0000903
     1'ONS ON I','TEMS IN','A DATA','SET (OR','FILE).','      '/                0000904
      TYPE 10, (WORDS(J),(TEXT(I,J),I=1,11),J=1,11)                             0000905
      RETURN                                                                    0000906
   10 FORMAT ('OTHE COMMANDS WHICH MAY BE ISSUED ','(AND THEIR MEANING)         0000907
     1ARE LISTED BELOW:'/(' ',A4,8A8/7X,3A8))                                   0000908
      END                                                                       0000909
```

FUNCTION NAME: ICONV

Purpose: ICONV is used to convert a number from unpacked-character form to numeric fixed-point form.

Calling sequence: IVAL = ICONV (TANK,LNGTH,EXP, ERR)

Arguments:

TANK—Contains the number to be converted in unpacked-character form.

LNGTH—The number of elements in TANK.

EXP—The power of 10 to which the value returned must be raised to obtain the floating point value represented.

ERR—An error flag which is turned on if an error is detected.

Subroutines called: None

Common data referenced: None

Called by: OPREP, PARSE, RELEXP

Error checking and reporting: Each character is checked. If an invalid character is detected, an error message is typed, and the error flag is turned on.

Program logic: If all characters are blank, blank is returned as a value. Leading blanks and imbedded commas are ignored, and a blank acts as a string deliminator. If a decimal point is sensed, the position counter EXP is initialized to 1 and incremented for each subsequent digit. As each digit is detected in a left-to-right fashion, the value is shifted left one digit, and the detected digit is added to the least significant part of the value. If all (LNGTH) characters have been scanned or if a trailing blank is detected, the scan is terminated and control is returned to the caller.

G R A S P S O U R C E P R O G R A M

```
      INTEGER FUNCTIONICONV(TANK,LNGTH,EXP,ERR)                          0000910
      INTEGER TANK( 1),DIGITS(10),BLANK,COMMA,POINT,EXP,VALUE            0000911
      LOGICAL ERR                                                        0000912
      DATA BLANK,COMMA,POINT/4H     ,4H,    ,4H.    /,DIGITS/4H0    ,4H1 0000913
     14H2    ,4H3    ,4H4    ,4H5    ,4H6    ,4H7    ,4H8    ,4H9    /   0000914
      ERR=.FALSE.                                                        0000915
      EXP=0                                                              0000916
      IF (LNGTH.GT.0) GO TO 10                                           0000917
      ICONV=BLANK                                                        0000918
      GO TO 90                                                           0000919
   10 VALUE=0                                                            0000920
      DO 20 K=1,LNGTH                                                    0000921
      IF (TANK(K).NE.BLANK) GO TO 30                                     0000922
   20 CONTINUE                                                           0000923
      GO TO 80                                                           0000924
   30 DO 70 J=K,LNGTH                                                    0000925
      NEXT=TANK(J)                                                       0000926
      IF (NEXT.EQ.COMMA) GO TO 70                                        0000927
      IF (NEXT.EQ.BLANK) GO TO 80                                        0000928
      IF (NEXT.NE.POINT) GO TO 40                                        0000929
      EXP=1                                                              0000930
      GO TO 70                                                           0000931
   40 IF (EXP.GT.0) EXP=EXP+1                                            0000932
      DO 50 L=1,10                                                       0000933
      IF (NEXT.EQ.DIGITS(L)) GO TO 60                                    0000934
   50 CONTINUE                                                           0000935
      TYPE 100, VALUE,EXP,(TANK(L),L=1,LNGTH)                            0000936
      ERR=.TRUE.                                                         0000937
      GO TO 90                                                           0000938
   60 VALUE=10*VALUE+L-1                                                 0000939
   70 CONTINUE                                                           0000940
   80 EXP=MINO(0,1-EXP)                                                  0000941
      ICONV=VALUE                                                        0000942
   90 RETURN                                                             0000943
  100 FORMAT (15H BAD CHARACTER ,I11,I5,3X,64A1)                         0000944
      END                                                                0000945
```

SUBROUTINE NAME: IFILE

Purpose: This subroutine is used to associate dynamically the name of an existing data set with a FORTRAN input unit number (that is, logical device number). When this subroutine is called, the file is opened and read; statements referencing the unit number given in the argument list are directed to the named file. The file may be closed by use of a rewind statement.

Calling sequence: CALL IFILE(I,NAME)

Arguments:

I—An integer variable or constant specifying a logical device number.

NAME—Either a literal (hollerith) constant or a variable containing a file name consisting of five or fewer characters.

Subroutines called: None

Common data referenced: None

Called by: BINTYP, DEFLST, FINDGP, OBEY, START, FILES

Error checking and reporting: None

Program logic: This is a DEC 1070, TOPS–10 system resident routine. It provides the capabilities referred to in the Purpose section above. If the GRASP system is to be implemented on some other main frame, a comparable routine must be written or acquired. No listing is shown here.

SUBROUTINE NAME: INIT

Purpose: INIT initializes a set of words and codes for future table lookup. The initialization assumes that the "linear quotient hash code" technique will be used for table lookup. See Bell and Kaman (1970) for a complete description of the technique.

Calling sequence: CALL INIT(CHARS,CODES,M,SYMBOL, N)

Arguments:

CHARS—Table which is to contain the symbols in "hash-coded" order.

CODES—Table giving the index of the symbol as it was stored in CHARS.

M—Table size for CHARS and CODES. Note this must be a prime number.

SYMBOL—Table of words to "hash" into CHARS.

N—Number of words in SYMBOL.

Subroutines called: None

Common data referenced: None

Called by: LOGEXP, PARSE

Error checking and reporting: None

Program logic:

1. CHARS is filled with "VOID," the flag for an empty location.
2. Each element of SYMBOL is then inserted into CHARS at the "hashed" address. "Collisions" are handled via the linear-quotient method.
3. As each element of SYMBOL is "hashed" into CHARS, its position is stored in CODES.
4. The initial probe address and collision displacement are determined from the contents of each element of SYMBOL.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE INIT(CHARS,CODES,M,SYMBOL,N)                    0000946
      INTEGER FILLER,CHARS(1),CODES(1),SYMBOL(1)                 0000947
      DATA FILLER/'VOID'/                                        0000948
      DO 10 I=1,M                                                0000949
   10 CHARS(I)=FILLER                                            0000950
      DO 40 ICODE=1,N                                            0000951
      ICHAR=SYMBOL(ICODE)                                        0000952
      L=IABS(ICHAR)                                              0000953
      J=L/M                                                      0000954
      I=L-M*J                                                    0000955
      IF (MOD(J,M).EQ.0) J=1                                     0000956
   20 IF (CHARS(I+1).EQ.FILLER) GO TO 30                         0000957
      I=MOD(I+J,M)                                               0000958
      GO TO 20                                                   0000959
   30 CHARS(I+1)=ICHAR                                           0000960
   40 CODES(I+1)=ICODE                                           0000961
      RETURN                                                     0000962
      END                                                        0000963
```

SUBROUTINE NAME: KEYBRD

Purpose: KEYBRD accepts all user input in unpacked character form and returns it to the caller.

Calling sequence: CALL KEYBRD(&m,IMAGE,N)

Arguments:

m—Statement number (in caller) to which a branch is made if an EOF is found.

IMAGE—Contains the user input in unpacked character form.

N—Number of characters to be read.

Subroutines called: None

Common data referenced: None

Called by: BDEF, COLPNT, CONDTN, DEFLST, FTNC, LIST, LOGEXP, OPREP, QUIT, RETRVE, VLIST, FILES, DRIVER, OBEY, PAUSE

Error checking and reporting: None

Program logic: KEYBRD accepts input from the user (unit 5) and takes nonstandard return if EOF occurs.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE KEYBRD(*,IMAGE,N)                                    0000964
      DATA IX/' '/                                                    0000965
      DIMENSION IMAGE(1)                                              0000966
      READ(5,20,END=10) (IMAGE(I),I=1,N)                              0000967
      IF(IMAGE(1).EQ.IX) GO TO 10                                     0000968
      RETURN                                                          0000969
   10 RETURN 1                                                        0000970
   20 FORMAT(80A1)                                                    0000971
      END                                                             0000972
```

SUBROUTINE NAME: LENGTH

Purpose: LENGTH determines the number of leading nonblank words in a character string.

Calling sequence: CALL LENGTH(VECT,N,L)

Arguments:

VECT—Array containing the character string to be examined.

N—The number of words to check.

L—The number of nonblank leading words.

Subroutines called: None

Common data referenced: None

Called by: DEFLST, DUMPIT, NAME

Error checking and reporting: None

Program logic: The first full-word blank is searched for; its position is returned in L.

GRASP SOURCE PROGRAM

```
      SUBROUTINE LENGTH(VECT,N,L)                                    0000973
      INTEGER VECT(1)                                                0000974
      DATA IBLNK/' '/                                                0000975
      L=0                                                            0000976
      DO 10 I=1,N                                                    0000977
      IF(VECT(I).EQ.IBLNK) GO TO 20                                  0000978
10      L=L+1                                                        0000979
20      RETURN                                                       0000980
      END                                                            0000981
```

SUBROUTINE NAME: LIST

Purpose: LIST is used as a driver for the listing of items from a selected file. LIST performs initialization common to both row and column forms of printout, and rewinds the input file after returning from the routine which created the printout.

Calling sequence: CALL LIST(&n)

Argument:

n—Statement (in caller) that is branched to if an EOF is sensed in any of the called routines.

Subroutines called: OPREP, KEYBRD, ROWPNT, COLPNT

Common data referenced: None

Called by: DRIVER

Error checking and reporting: The user response returned by KEYBRD is checked for validity. If in error, a message is typed, and the user is prompted again.

Program logic:

1. The input file name and page size are set by a call to OPREP.
2. The user is prompted for "C" (column) or "R" (row) to establish the desired output form.
3. Either ROWPNT or COLPNT is called depending on the user's response to step 2 above.
4. The input file is rewound prior to the return to DRIVER.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE LIST(*)                                                      0000982
      COMMON /IOUNIT/ NFILE,IOFILE                                            0000983
      DATA IC,IR/'C','R'/                                                     0000984
      CALL OPREP(&50,&45,NPAGE)                                               0000985
   10 TYPE 60                                                                 0000986
      CALL KEYBRD(&50,IM,1)                                                   0000987
      IF (IM.EQ.IC) GO TO 30                                                  0000988
      IF (IM.EQ.IR) GO TO 20                                                  0000989
      TYPE 70                                                                 0000990
      GO TO 10                                                                0000991
   20 TYPE 80                                                                 0000992
      CALL ROWPNT(&50,NPAGE)                                                  0000993
      GO TO 40                                                                0000994
   30 TYPE 80                                                                 0000995
      CALL COLPNT(&50,NPAGE)                                                  0000996
   40 REWIND NFILE                                                            0000997
45      RETURN                                                               0000998
   50 REWIND NFILE                                                            0000999
      RETURN 1                                                                0001000
   60 FORMAT (' ENTER C FOR COLUMN OR R FOR ROW PRINTING: ',$)                0001001
   70 FORMAT (' YOUR REPLY WAS NOT UNDERSTOOD.')                              0001002
   80 FORMAT (' AT EACH PAUSE PRESS CR KEY TO CONTINUE. ','TO ABORT ENTE0001003
     1R A.')                                                                  0001004
      END                                                                     0001005
```

SUBROUTINE NAME: LOGEXP

Purpose: This routine accepts a logical expression as user input via a call to KEYBRD and returns the encoded Reverse-Polish form of the expressions. The logical expression may be composed of single-letter (A–Z) operands which refer to previously entered conditions, the logical operators "and," "or," "not," and the grouping symbols (,). Each of the logical operators may be denoted in two ways, as follows: .AND. or *, .OR. or +, .NOT. or —.

Calling sequence: CALL LOGEXP(&n,POLISH,LPS,NCOND)

Arguments:

n—Statement number (in calling routine) to which a branch will be made if an EOF is sensed by KEYBRD.

POLISH—Contains the encoded Reverse-Polish form of a logical expression. Let n denote the value of some element of POLISH. Then $1 \leq n \leq 26$ implies reference to the nth condition entered. If $29 \leq n \leq 31$, the logical operators OR, AND, NOT correspond to these three values. No other values will be assumed by elements of POLISH.

LPS—Gives the number of elements in POLISH.

NCOND—Gives the number of conditions which have been entered by a previous call to CONDTN.

Subroutines called: INIT, KEYBRD, SCAN, FIND, PACK

Common data referenced: LOGIC in /INPUT/

Called by: DRIVER

Error checking and reporting: The logic expression entered is checked for syntactic correctness. Following are eight error messages which may be typed:

1. LOGICAL OPERATOR NOT PRECEDED BY A) OR A LETTER (A–Z).
2. UNBALANCED PARENTHESIS.
3. LETTER (A–Z) NOT SEPARATED BY AN OPERATOR.
4. UNEXPECTED LEFT PARENTHESIS OR .NOT. OPERATOR (—).
5. INVALID CHARACTER IN EXPRESSION.
6. UNDETERMINED SYNTAX ERROR. CONTACT PROGRAMMER.
7. LOGIC EXPRESSION REFERENCES A CONDITION (A–Z) WHICH WAS NOT ENTERED.
8. OPERATOR NOT ENCLOSED WITH PERIODS. RE-ENTER LOGIC.

Program logic:

1. On the first call to LOGEXP, a call to INIT is made to "hash-code" the elements of SYMBOL into CHARS. CODES is used to save the original indices.
2. A prompt message is typed and a call to KEYBRD is made to get the input string which is then packed into LOGIC.
3. After initialization of pointers and counters, a call is made to SCAN to bracket the nonblank section of STRING.
4. At this point the actual algorithm begins. Transition matrix parsing is used with the following transition matrix (IMAT):

	A–Z	+ *	(—	)	blank	.
1.	f(1)/2	f(7)/1	f(4)/1	f(7)/2	f(2)/1	f(6)/1
2.	f(7)/3	f(3)/1	f(7)/4	f(5)/2	f(2)/2	f(6)/1

where f(i)/j means "do the i'th job and set the next row value to j." The jobs are:

f(1)—Insert character code into Reverse-Polish string and test to determine if there has been a condition entered for it.

f(2)—Go scan next character.

f(3)—Pop stack into POLISH until value of topmost element is less than character code. Then do f(4).

f(4)—Push down character code into stack.

f(5)—Pop stack into POLISH until the value for (is reached. Remove value for (.

f(6)—Period character sensed, find next matching period and determine logical operator.

f(7)—Type the error message pointed to by the row value, then request reentry of logic.

Each character of STRING is scanned using the subroutine FIND to obtain its code ICODE. ICODE is an index to ICOLS which then determines the proper column of IMAT. This element IFTN is then broken down into a function pointer JOB and a next row value STATE. Control is then passed to the function indicated by JOB. After the function has been completed, the next character of STRING is scanned if an error was not detected. If an error was detected, the appropriate message is typed.

GRASP SOURCE PROGRAM

```
      SUBROUTINE LOGEXP(*,POLISH,LPS,NCOND)                              0001006
      COMMON /INPUT/ EXPR,LOGIC                                          0001007
      DOUBLE PRECISION ERRMSG(7,6),EXPR(4,26),LOGIC(8)                   0001008
      INTEGER POLISH(1),ICOLS(33),CHARS(41),TOP,FC,STATE,OR,AND,STACK(1  0001009
     15),IMAT(2,6),CODES(41),SYMBOL(33),STRING(80),PERIOD,BLANK          0001010
      LOGICAL CALLED                                                     0001011
      DATA SYMBOL/'A','B','C','D','E','F','G','H','I','J','K','L','M','N0001012
     1','O','P','Q','R','S','T','U','V','W','X','Y','Z','(',')','+','*',0001013
     2'-',' ','.'/                                                       0001014
      DATA ICOLS/26*1,3,4,2,2,3,5,6/, CALLED/.FALSE./                    0001015
      DATA IMAT/12,73,71,31,41,74,72,52,21,22,61,61/                     0001016
      DATA ERRMSG/'LOGICAL','OPERATOR',' NOT PRE','CEDED BY',' A ) OR',  0001017
     1'A LETTER',' (A-Z).','UNBALANC','ED PAREN','THESIS.',4*'      ',   0001018
     2'LETTER (','A-Z) NOT',' SEPARAT','ED BY AN',' OPERATO','R.    ',   0001019
     3'      ','UNEXPECT','ED LEFT','PARENTHE','SIS OR .','NOT. OPE',   0001020
     4'RATOR (-',').    ','INVALID','CHARACTE','R IN EXP','RESSION.',   0001021
```

```
     53*'      ','UNDETERM','INED SYN','TAX ERRO','R. CONTA','CT PROGR',0001022
     6'AMMER.','      '/,OR,AND,NOT/'OR','AND','NOT'/                    0001023
      EQUIVALENCE (SYMBOL(32),BLANK),(SYMBOL(33),PERIOD)                0001024
      IF (CALLED) GO TO 20                                              0001025
      CALLED=.TRUE.                                                     0001026
      CALL INIT(CHARS,CODES,41,SYMBOL,33)                               0001027
   20 TYPE 250                                                          0001028
      CALL KEYBRD(&245,STRING,80)                                       0001029
      CALL PACK(STRING,LOGIC,80,80)                                     0001030
      TOP=1                                                             0001031
      STACK(1)=0                                                        0001032
      LPS=0                                                             0001033
      STATE=1                                                           0001034
      CALL SCAN(&30,STRING,1,IPT,LNGTH,2)                               0001035
      IPT=IPT-1                                                         0001036
      GO TO 40                                                          0001037
30      RETURN                                                          0001038
   40 NSTATE=STATE                                                      0001039
      IPT=IPT+1                                                         0001040
      IF (IPT.GT.LNGTH) GO TO 230                                       0001041
      CALL FIND(&50,STRING(IPT),ICODE,CHARS,CODES,41)                   0001042
      GO TO 60                                                          0001043
   50 STATE=5                                                           0001044
      GO TO 220                                                         0001045
   60 IFTN=IMAT(NSTATE,ICOLS(ICODE))                                    0001046
      JOB=IFTN/10                                                       0001047
      STATE=IFTN-10*JOB                                                 0001048
      GO TO (70,40,90,100,110,130,220),JOB                              0001049
   70 LPS=LPS+1                                                         0001050
      POLISH(LPS)=ICODE                                                 0001051
      IF (ICODE.LE.NCOND) GO TO 40                                      0001052
      TYPE 270                                                          0001053
      GO TO 20                                                          0001054
   90 IF (STACK(TOP).LT.ICODE) GO TO 100                                0001055
      LPS=LPS+1                                                         0001056
      POLISH(LPS)=STACK(TOP)                                            0001057
      TOP=TOP-1                                                         0001058
      IF (TOP.GT.0) GO TO 90                                            0001059
      STATE=6                                                           0001060
      GO TO 220                                                         0001061
  100 TOP=TOP+1                                                         0001062
      STACK(TOP)=ICODE                                                  0001063
      GO TO 40                                                          0001064
  110 IF (STACK(TOP).EQ.27) GO TO 120                                   0001065
      LPS=LPS+1                                                         0001066
      POLISH(LPS)=STACK(TOP)                                            0001067
      TOP=TOP-1                                                         0001068
      IF (TOP.GT.0) GO TO 110                                           0001069
      STATE=2                                                           0001070
      GO TO 220                                                         0001071
  120 TOP=TOP-1                                                         0001072
      IF (TOP.GT.0) GO TO 40                                            0001073
      STATE=2                                                           0001074
      GO TO 220                                                         0001075
  130 FC=IPT+1                                                          0001076
      DO 140 I=FC,80                                                    0001077
      IF (STRING(I).NE.BLANK) GO TO 160                                 0001078
  140 CONTINUE                                                          0001079
  150 TYPE 280                                                          0001080
      GO TO 20                                                          0001081
```

```
160 FC=I                                                                0001082
    DO 170 IPT=FC,80                                                    0001083
    IF (STRING(IPT).EQ.PERIOD) GO TO 180                                0001084
170 CONTINUE                                                            0001085
    GO TO 150                                                           0001086
180 NCHAR=3                                                             0001087
    ICODE=0                                                             0001088
190 NCHAR=NCHAR-1                                                       0001089
    NCH=FC+NCHAR                                                        0001090
    IOP=CHARS(21)                                                       0001091
    CALL PACK(STRING(FC),IOP,NCH-FC+1,4)                                0001092
    IF (IOP.EQ.AND) ICODE=30                                            0001093
    IF (IOP.EQ.NOT) ICODE=31                                            0001094
    IF (IOP.EQ.OR) ICODE=29                                             0001095
    IF (ICODE.NE.0) GO TO 200                                           0001096
    IF (NCHAR.GT.1) GO TO 190                                           0001097
    STATE=6                                                             0001098
    GO TO 220                                                           0001099
200 IF (ICODE.LT.31) GO TO 210                                          0001100
    IF (NSTATE.EQ.1) GO TO 100                                          0001101
    STATE=4                                                             0001102
    GO TO 220                                                           0001103
210 IF (NSTATE.EQ.2) GO TO 90                                           0001104
    STATE=1                                                             0001105
220 TYPE 300, (ERRMSG(I,STATE),I=1,7)                                   0001106
    GO TO 20                                                            0001107
230 IF (TOP.EQ.1) GO TO 30                                              0001108
    IF (STACK(TOP).GT.28) GO TO 240                                     0001109
    STATE=2                                                             0001110
    GO TO 220                                                           0001111
240 LPS=LPS+1                                                           0001112
    POLISH(LPS)=STACK(TOP)                                              0001113
    TOP=TOP-1                                                           0001114
    GO TO 230                                                           0001115
245 RETURN 1                                                            0001116
250 FORMAT (' ENTER LOGIC: ',$)                                         0001117
270 FORMAT (' LOGIC EXPRESSION REFERENCES A CONDITION (A-Z)',' WHICH W0001118
   1AS NOT ENTERED.')                                                   0001119
280 FORMAT (' OPERATOR NOT ENCLOSED WITH PERIODS. RE-ENTER LOGIC.')     0001120
300 FORMAT (' LOGICAL ERROR:'/1X,7A8)                                   0001121
    END                                                                 0001122
```

SUBROUTINE NAME: MEAN

Purpose: MEAN provides for the computation of range, mean, sum, root mean square, and sum of squares for as many as five specified items in a specified file.

Calling sequence: CALL MEAN(J,ISWTCH)

Arguments:

J—Pointer used to retrieve argument values from the common area /FTNCOM/.

ISWTCH—Switch indicating which of three parts (initialization, body, postprocessing) of the code is to be executed.

Subroutine called: UNCODE

Common data referenced:

ITYPE in blank common

TAGS, IREC, ARGS, NARGS in /FTNCOM/

Called by: FDRIVE

Error checking and reporting: None

Program logic: The value of ISWTCH determines which of three sections of the code is executed.

If ISWTCH=1, sums and range values are initialized. If ISWTCH=2, the type for each argument value is determinted and its value is added to the appropriate sums. Range values are updated if required. If ISWTCH=3, the final computations are performed and the results typed out to the user.

The mean is determined using $\overline{X}=\Sigma X/N$ and the root mean square is determined using $RMS=\Sigma X^2/N$, where N is the number of nonblank values of X.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE MEAN(J,ISWTCH)                                                  0001123
      COMMON NAMES,ITYPE,IPTS,IDIM                                               0001124
      COMMON /FTNCOM/ TAGS,IREC,ARGS,NARGS,IFTN,NFTN                             0001125
      DIMENSION ARGS(6,5), NARGS(5), IFTN(5), ITYPE(500), IREC(500),             0001126
     1IPTS(500),NSUM(5), SUMX(5), SUMXS(5), VMAX(5), VMIN(5)                     0001127
      DOUBLE PRECISION NAMES(500),TAGS(5,5)                                      0001128
      INTEGER ARGS                                                               0001129
      EQUIVALENCE (IVAL,VAL)                                                     0001130
      DATA IBLNK/' '/                                                            0001131
      IF(ISWTCH-2) 5,15,25                                                       0001132
    5 K=NARGS(J)                                                                 0001133
      DO 10 I=1,K                                                                0001134
      SUMX(I)=0.                                                                 0001135
      VMAX(I)=-1.E30                                                             0001136
      NSUM(I)=0                                                                  0001137
      VMIN(I)=1.E30                                                              0001138
   10 SUMXS(I)=0.                                                                0001139
      GO TO 55                                                                   0001140
   15 K=NARGS(J)                                                                 0001141
      DO 20 I=1,K                                                                0001142
      IVAL=IREC(ARGS(I+1,J))                                                     0001143
      IF (IVAL.EQ.IBLNK) GO TO 20                                                0001144
      NSUM(I)=NSUM(I)+1                                                          0001145
      VALUE=IVAL                                                                 0001146
      IF(ITYPE(ARGS(I+1,J)).EQ.5) VALUE=UNCODE(VAL,IQ)                           0001147
      IF (ITYPE(ARGS(I+1,J)).EQ.2) VALUE=VAL                                     0001148
      IF (VALUE.LT.VMIN(I)) VMIN(I)=VALUE                                        0001149
      IF (VALUE.GT.VMAX(I)) VMAX(I)=VALUE                                        0001150
      SUMX(I)=SUMX(I)+VALUE                                                      0001151
      SUMXS(I)=SUMXS(I)+VALUE*VALUE                                              0001152
   20 CONTINUE                                                                   0001153
      GO TO 55                                                                   0001154
   25 K=NARGS(J)                                                                 0001155
      DO 50 I=1,K                                                                0001156
      IF (NSUM(I).EQ.0) GO TO 30                                                 0001157
      TYPE 60, TAGS(I,J),NSUM(I)                                                 0001158
      AMEAN=SUMX(I)/NSUM(I)                                                      0001159
      RMS=SUMXS(I)/NSUM(I)                                                       0001160
      IF (ITYPE(ARGS(I+1,J)).EQ.2) GO TO 40                                      0001161
      IF (ITYPE(ARGS(I+1,J)).EQ.5) GO TO 40                                      0001162
```

```
      MIN=VMIN(I)                                                          0001163
      MAX=VMAX(I)                                                          0001164
      TYPE 80, MIN,MAX,AMEAN,RMS,SUMX(I),SUMXS(I)                          0001165
      GO TO 50                                                             0001166
   30 TYPE 70, TAGS(I,J)                                                   0001167
      GO TO 50                                                             0001168
   40 TYPE 80, VMIN(I),VMAX(I),AMEAN,RMS,SUMX(I),SUMXS(I)                  0001169
   50 CONTINUE                                                             0001170
   55 RETURN                                                               0001171
   60 FORMAT (/' MEAN STATISTICS FOR ',A8,'WITH',I6,' ITEM(S).')           0001172
   70 FORMAT (/' NO VALUES PRESENT FOR ',A8)                               0001173
   80 FORMAT ('  MIN=',1PG9.2,'  MAX=',G9.2,'  MEAN=',G9.2,'  ROOT MEAN   0001174
     1SQ.=',G9.2/'  SUM=',G12.5,'  SUM OF SQUARES=',G12.5)                 0001175
      END                                                                  0001176
```

SUBROUTINE NAME: NAME

Purpose: NAME provides the user with the mechanism for examining the structure and content of the current data base. The user is permitted to select categories of interest. Item names, types, and descriptions for all entries in selected categories are printed. The user is also permitted to see the values which are assumed (in the current data base) by character- and multiple-choice-type items.

Calling sequence: CALL NAME (&n)

Argument:

n—Statement number (in caller) to which a branch is made if the second nonstandard return is taken from DEFLST (EOF sensed by KEYBRD), or EOF is sensed in BDEF.

Subroutines called: DEFLST, PAUSE, FINGDP, LENGTH, BDEF

Common data referenced: None

Called by: DRIVER

Error checking and reporting: None

Program logic:

1. DEFLST is called to select those categories of interest (LIST).
2. For each category selected, a call is made to FINDGP to obtain the names, types, and descriptions for all items within that category. They are then printed under the category name.
3. Programmed pauses after each category or 30 lines of output are provided via calls to PAUSE.
4. A call to LENGTH is made to determine the number of nonblank words in the description.
5. Unit 20 (used by DEFLST and FINDGP) is rewound prior to returning to DRIVER.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE NAME(*)                                                        0001177
      INTEGER CAT(8,17),DESC(12,45),TYPE(45),LIST(17)                           0001178
      DOUBLE PRECISION NAMES(45)                                                0001179
      CALL DEFLST(&50,&60,CAT,NUMC,MC,LIST)                                     0001180
      TYPE 70                                                                   0001181
      CALL PAUSE(&50)                                                           0001182
      DO 30 K=1,NUMC                                                            0001183
      KNUM=LIST(K)                                                              0001184
      CALL FINDGP(&50,KNUM,NUM,MAXL,NG,DESC)                                    0001185
      READ (20) (NAMES(J),TYPE(J),(DESC(I,J),I=1,MAXL),J=1,NUM)                 0001186
      CALL LENGTH(CAT(1,KNUM),MC,MCL)                                           0001187
      TYPE 90, (CAT(I,KNUM),I=1,MCL)                                            0001188
      TYPE 91                                                                   0001189
      LINE=0                                                                    0001190
   10 LINE=LINE+1                                                               0001191
      IF (LINE.GT.NUM) GO TO 30                                                 0001192
      IF (MOD(LINE,30).NE.0) GO TO 20                                           0001193
      CALL PAUSE(&50)                                                           0001194
      TYPE 90, (CAT(I,KNUM),I=1,MCL)                                            0001195
      TYPE 100                                                                  0001196
      TYPE 91                                                                   0001197
20      CALL LENGTH(DESC(1,LINE),MAXL,MXL)                                      0001198
       TYPE 110, NAMES(LINE),TYPE(LINE),(DESC(I,LINE),I=1,MXL)                  0001199
      GO TO 10                                                                  0001200
   30 CALL PAUSE(&50)                                                           0001201
   50 REWIND 20                                                                 0001202
      CALL BDEF(&60)                                                            0001203
      RETURN                                                                    0001204
   60 RETURN 1                                                                  0001205
   70 FORMAT (' IN EACH CATEGORY, THE ITEM NAMES, TYPE CODES,',' AND DES0001206
     1CRIPTIONS WILL BE'/' LISTED. TYPE CODES:'/9X,'I = WHOLE NUMBERS'/90001207
     2X,'R = NUMBERS WITH FRACTIONAL PARTS'/9X,'A = ALPHANUMERIC STRINGS0001208
     3'/9X,'B = MULTIPLE CHOICE TYPES'/9X,'Q = QUALIFIED NUMERIC VALUES'0001209
     4 /' AT EACH PAUSE STRIKE CR KEY TO CONTINUE (STARTING NOW).')      0001210
90      FORMAT(' CATEGORY: ',9A5)                                                 0001211
91      FORMAT('  NAME TYPE  DESCRIPTION'/'  ----',                               0001212
     1 ' ----  -----------')                                                      0001213
100     FORMAT('+',T50,'(CON''T)')                                                0001214
  110 FORMAT (1X,A7,1X,A1,2X,12A5)                                              0001215
      END                                                                       0001216
```

SUBROUTINE NAME: OBEY

Purpose: OBEY associates input or output file names with FORTRAN unit numbers and provides a degree of file-name checking and protection.

Calling sequence: CALL OBEY(&m,MSG,N)

Arguments:

m—Statement (in caller) to which a branch will be made if a protected (output) or unknown (input) file name is referenced.

MSG—Contains (in packed-character form) either of the following:
"EQUATE 11 name," indicating input,
"EQUATE 12 name," indicating output.

N—The number of words in MSG.

Subroutines called: KEYBRD, IFILE, OFILE

Common data referenced:

IN, IOUT in /IOUNIT/
FNAMES, NUMI, WHICH in /FILNAM/

Called by: RETRVE, OPREP, FTNC

Error checking and reporting:

1. Input file names are checked for recognition.
2. Output file names are checked to prevent writing on a "protected" file.
3. The total number of output files is checked against the maximum 20.

   Error messages for each of the above three checks are provided.

Program logic:

1. If the message length N is not 4, or if the first 4 characters in MSG are not EQUA, return is immediate.
2. MASTER is set to the current data-base name in /FILNAM/ and MSG is moved to IMSG.
3. If the third word of IMSG is not "2," input is assumed, and the file name (last word of IMSG) is checked against the names of files created during this session. If no match is found, the user is informed and given the opportunity to exit and enter a new command. Finally, IFILE is called to associate unit 23 with the file name provided.
4. If the third word of IMSG is "2," output is assumed, and the file name FNAME (last word in IMSG) is checked against the list of protected file names in /FILNAM/. If a match is found, the nonstandard return is taken. Otherwise, FNAME is added to the list FILES, and unit 24 is associated with the file FNAME via a call to OFILE.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE OBEY(*,MSG,N)                                          0001217
      COMMON /IOUNIT/ IN,IOUT                                           0001218
      COMMON /FILNAM/ FNAMES,NUMI,WHICH,PAD                             0001219
      INTEGER MASTER,FNAME,FILES(20),MSG(1),IMSG(4),FNAMES(20),WHICH    0001220
     1 ,PAD(4)                                                          0001221
      EQUIVALENCE (IMSG(4),FNAME)                                       0001222
      DATA I2,NUMF,IEQ/'2',0,'EQUA'/,IYES/'Y'/                          0001223
      IF(N.NE.4) GO TO 100                                              0001224
      IF(MSG(1).NE.IEQ) GO TO 100                                       0001225
      MASTER=FNAMES(WHICH)                                              0001226
      DO 2 I=1,4                                                        0001227
    2 IMSG(I)=MSG(I)                                                    0001228
      IF(IMSG(3).EQ.I2) GO TO 10                                        0001229
      IN=23                                                             0001230
      IF(FNAME.EQ.MASTER) GO TO 6                                       0001231
      IF(NUMF.GT.0) GO TO 4                                             0001232
    3 TYPE 500                                                          0001233
      CALL KEYBRD(&100,I,1)                                             0001234
      IF(I.EQ.IYES) GO TO 6                                             0001235
    8  RETURN 1                                                         0001236
    4 DO 5 I=1,NUMF                                                     0001237
      IF(FILES(I).EQ.FNAME) GO TO 6                                     0001238
    5 CONTINUE                                                          0001239
      GO TO 3                                                           0001240
    6 CALL IFILE(IN,FNAME)                                              0001241
      GO TO 100                                                         0001242
   10 DO 11 J=1,5                                                       0001243
      K=4*(J-1)                                                         0001244
      DO 11 I=1,NUMI                                                    0001245
      IF(FNAMES(K+I).EQ.FNAME) GO TO 14                                 0001246
   11 CONTINUE                                                          0001247
      GO TO 12                                                          0001248
```

```
   14 TYPE 501,FNAME                                                    0001249
      GO TO 8                                                           0001250
   12 IF(NUMF.EQ.0) GO TO 20                                            0001251
      DO 13 I=1,NUMF                                                    0001252
      IF(FILES(I).EQ.FNAME) GO TO 21                                    0001253
   13 CONTINUE                                                          0001254
   20 NUMF=NUMF+1                                                       0001255
      IF(NUMF.LE.20) GO TO 22                                           0001256
      NUMF=20                                                           0001257
      TYPE 23                                                           0001258
      GO TO 8                                                           0001259
   22 FILES(NUMF)=FNAME                                                 0001260
   21 IOUT=24                                                           0001261
      CALL OFILE(IOUT,FNAME)                                            0001262
  100 RETURN                                                            0001263
   23 FORMAT(' NO MORE THAN 20 FILES MAY BE CREATED IN ONE RUN.')       0001264
500     FORMAT(' ATTEMPT TO REFERENCE A FILE NOT CREATED THIS RUN.'/     0001265
     1 ' DO YOU STILL WANT IT? (Y OR N): ',$)                           0001266
  501 FORMAT(1X,A6,'MAY NOT BE USED AS AN OUTPUT FILE NAME')            0001267
      END                                                               0001268
```

SUBROUTINE NAME: OFILE

Purpose: This subroutine is used to associate dynamically the name of a new data set with a FORTRAN output unit number (that is, logical device number). When this subroutine is called, the file is opened, and write statements referencing the unit number given in the argument list are directed to the named file. The file may be closed by use of a rewind statement.

Calling sequence: CALL OFILE(I,NAME)

Arguments:

I—An integer variable or constant specifying a logical device number.

NAME—Either a literal (hollerith) constant or variable containing a file name consisting of five or fewer characters.

Subroutines called: None

Common data referenced: None

Called by: COLPNT, OBEY

Error checking and reporting: None

Program logic: This routine is a DEC 1070, TOPS–10 system resident routine. It provides the capabilities referred to in the section "Purpose" above. If the GRASP system is to be implemented on some other main frame, a comparable routine must be written or acquired. Therefore, a listing has not been provided.

SUBROUTINE NAME: OPREP

Purpose: This routine is used to prompt for and accept the name of a file and a page size.
Calling sequence: CALL OPREP(&n,&m,NPAGE)
Arguments:
n—Statement number (in calling routine) to which a branch will be made if an EOF is sensed by KEYBRD.
m—Statement number (in calling routine) to which a branch will be made if the nonstandard return from OBEY is taken.
NPAGE—Page size (in lines) as entered by user.
Subroutines called: KEYBRD, OBEY, ICONV, PACK
Common data referenced: NAMES, WHICH in /FILNAM/
Called by: DUMPIT, LIST
Error checking and reporting: An error flag set by ICONV is tested. If set, the user is requested to reenter the value.
Program logic:
1. A file name is prompted for and accepted (via KEYBRD). This name is then packed into FNAME and compared with blank.
2. If blank, FNAME is set to the default file name obtained in /FILNAM/.
3. FNAME is then passed to OBEY via EQUATE.
4. A page size is then prompted for and accepted in character form. The numeric value is obtained by a reference to ICONV. If zero, then NPAGE is set to 10 million.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE OPREP(*,*,NPAGE)                                                    0001269
      COMMON /FILNAM/ FNAMES,WHICH,PAD                                               0001270
      INTEGER DBLNK,DFAULT,FNAME,EQUATE(4),TANK(5),FNAMES(21),WHICH                  0001271
     1 ,PAD(4)                                                                       0001272
      LOGICAL BAD                                                                    0001273
      EQUIVALENCE (EQUATE(4),FNAME)                                                  0001274
      DATA DBLNK,EQUATE/' ','EQUA','TE 1','1',' '/                                   0001275
      TYPE 20                                                                        0001276
      DFAULT=FNAMES(WHICH)                                                           0001277
      CALL KEYBRD(&15,TANK,5)                                                        0001278
      CALL PACK(TANK,FNAME,5,5)                                                      0001279
      IF(FNAME.EQ.DBLNK) FNAME=DFAULT                                                0001280
      CALL OBEY(&18,EQUATE,4)                                                        0001281
   10 TYPE 30                                                                        0001282
      CALL KEYBRD(&15,TANK,5)                                                        0001283
      NPAGE=ICONV(TANK,5,I,BAD)                                                      0001284
      IF (BAD) GO TO 10                                                              0001285
      IF (NPAGE.EQ.0) NPAGE=10000000                                                 0001286
      RETURN                                                                         0001287
   15 RETURN 1                                                                       0001288
18       RETURN 2                                                                   0001289
   20 FORMAT (' ENTER NAME OF FILE: ',$)                                             0001290
   30 FORMAT (' ENTER NUMBER OF LINES/PAGE: ',$)                                     0001291
      END                                                                            0001292
```

SUBROUTINE NAME: PACK

Purpose: All user input to the GRASP system is in unpacked form (in other words, one single left-justified character per word). PACK is used to convert from this unpacked form to packed form. This is necessary because character data in files accessed by GRASP is in packed form to conserve space.

Calling sequence: CALL PACK(SOURCE,DESTN,N,SIZE)

Arguments:

SOURCE—The array containing the unpacked character string.

DESTN—The array which is to contain the packed character string.

N—The number of characters to pack.

SIZE—The size (in characters) of the area to receive the packed data.

Subroutines called: None

Common data referenced: None

Called by: DRIVER, COLPNT, CONDTN, DECOMP, FILES, FTNC, LOGEXP, OPREP, PARSE, PNTER, RELEXP, RETRVE, VLIST

Error checking and reporting: None

Program logic: The ENCODE statement is used to move the characters from the unpacked string (SOURCE) to the packed string (DESTN).

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE PACK(SOURCE,DESTN,N,SIZE)                    0001293
      INTEGER SOURCE(1),DESTN(1),SIZE                         0001294
      ENCODE(SIZE,1,DESTN) (SOURCE(I),I=1,N)                  0001295
    1 FORMAT(80A1)                                            0001296
      RETURN                                                  0001297
      END                                                     0001298
```

SUBROUTINE NAME: PARSE

Purpose: PARSE converts arithmetic expressions to an encoded Reverse-Polish form. Extensive syntax checking, conversion, and preliminary addressing are performed to facilitate later evaluation by EVAL. The arithmetic expressions may contain the usual arithmetic operators (+,—,*,/), numeric constants, item names, and the following functions:

ABS ( )—absolute value;
SQRT ( )—square root;
LOG ( )—log base 10;
SQR ( )—square;
TEN ( )—power of 10.

Parentheses may be used for grouping to control the order of evaluation.

Calling sequence: CALL PARSE(EXPR,L,TYPE,POLISH, I,ERR)

Arguments:

EXPR—Arithmetic expression to be parsed in unpacked character form.
L—The length of EXPR.
TYPE, POLISH—Arrays which will contain the encoded Reverse-Polish form. See section on subroutine EVAL for additional encoding information.
I—Length of TYPE and POLISH.
ERR—Logical flag set if an error is detected.

Subroutines called: INIT, FIND, BFIND, INCONV, PACK

Common data referenced: NAMES, IPNTS, IDIM in blank common.

Called by: PREVAL

Error checking and reporting: The expression is checked for normal FORTRAN-like syntax (such as balanced parentheses, binary operators bracketed by valid names or expressions, and correct spelling of function names). The message "ERROR IN EXPRESSION" is typed if an error is detected. If an operand or function name is not recognized, that message is typed.

Program logic: The logical variable CALLED is tested. If it has not been set by a previous call, it is set to .TRUE. and INIT is called to "hash code" the elements of SYMBOL into CHARS and CODES. Next, the variables ERR, ROW, TOP, I, and C are initialized. The remainder of the computation involves scanning EXPR, an element at a time. As in LOGEXP, a transition-matrix technique is used to parse the expression, converting it to reverse-Polish form. The transition matrix (TM) is given below:

	A–Z %	—	+	/	*	(	)	blank	0—9
1.	f(1)/2	f(8)/1	error	error	error	f(10)/1	error	f(2)/1	f(11)/3
2.	f(3)/2	f(4)/1	f(5)/1	f(6)/1	f(7)/1	f(12)/1	f(9)/2	f(2)/2	f(3)/2
3.	error	f(4)/1	f(5)/1	f(6)/1	f(7)/1	error	f(9)/3	f(2)/3	f(3)/3

where the f(i) are separate tasks as follows:

f(1)—Start a name.
f(2)—Go scan next character.
f(3)—Append current character to name.
f(4–7)—Binary arithmetic operator sensed: Set CODE to indicated operator; pop stack until CODE is less than the topmost stack element; push CODE down on stack.
f(8)—Unary minus sensed: set CODE and push down on stack.
f(9)—Right parenthesis sensed: pop stack until topmost element is code for left parenthesis (PAREN); decrease size of stack by one.
f(10)—Left parenthesis sensed; push down parenthesis code PAREN.
f(11)—Digit or period sensed: start a constant. NAME is used to contain the constant in character form.
f(12)—Left parenthesis sensed in row 2: hence, the contents of NAME are assumed to be a function name. Check for validity and print an error message if invalid; otherwise, set CODE and push down on stack.

The proper element of TM is selected by the variables ROW and COLUMN. The COLUMN value is determined by a lookup (via FIND) of the current character and the ROW value is set by the last element of TM referenced. Once the proper element of TM is selected, the next ROW value is set and a branch is made to the current task.

This process is repeated until all elements of EXPR have been processed. See the section on subroutine EVAL for details of the encoding of TYPE and POLISH.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE PARSE(EXPR,L,TYPE,POLISH,I,ERR)                        0001299
      COMMON NAMES,ITYPE,IPNTS,IDIM                                     0001300
      DIMENSION NAMES(500), POLISH(1), STACK(41), EXPR(1), NAME(8), TM(30001301
     1,9), SYMBOL(45), COLS(45), TYPE(1), ITYPE(500), IPNTS(500), IFNCTS0001302
     2(5), CHARS(47), CODES(47)
      DOUBLE PRECISION NAMES,VARBLE,DBLNK                               0001303
      INTEGER TM,TOP,ROW,COLUMN,ELEMNT,SWITCH,TYPE,C,COLS,SYMBOL,CHARS,C0001305
     1ODES,CHAR,EXPR
      LOGICAL POP,NUM,ERR,CALLED                                        0001306
      DATA TM/12,32,0,81,2*41,0,2*51,0,2*61,0,2*71,101,121,2*0,92,93,21,0001308
     122,23,113,32,33/
      DATA SYMBOL/'A','B','C','D','E','F','G','H','I','J','K','L','M','N0001310
     1','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3',0001311
     2'4','5','6','7','8','9','.',' ','+','-','*','/','(',')','%'/     0001312
      DATA COLS/26*1,11*9,8,3,2,5,4,6,7,1/                              0001313
      DATA IBLNK,CALLED,DBLNK/' ',.FALSE.,'      '/                     0001314
      DATA PAREN,PLUS,DIFF,PROD,DIV,UNARY/0.,-1.,-2.,-3.,-4.,-5./       0001315
```

```
      DATA IFNCTS/'ABS','SQRT','LOG','SQR','TEN'/                          0001316
      IF (CALLED) GO TO 20                                                 0001317
      CALLED=.TRUE.                                                        0001318
      CALL INIT(CHARS,CODES,47,SYMBOL,45)                                  0001319
   20 ERR=.FALSE.                                                          0001320
      ROW=1                                                                0001321
      TOP=0                                                                0001322
      I=0                                                                  0001323
      C=0                                                                  0001324
   30 C=C+1                                                                0001325
      IF (C.LE.L) GO TO 50                                                 0001326
      IF (ROW.EQ.1) GO TO 350                                              0001327
      IF (POP) GO TO 250                                                   0001328
      IF (NUM) GO TO 40                                                    0001329
      SWITCH=1                                                             0001330
      GO TO 300                                                            0001331
   40 SWITCH=2                                                             0001332
      GO TO 320                                                            0001333
   50 CHAR=EXPR(C)                                                         0001334
      CALL FIND(&350,CHAR,COLUMN,CHARS,CODES,47)                           0001335
      COLUMN=COLS(COLUMN)                                                  0001336
      ELEMNT=TM(ROW,COLUMN)                                                0001337
      JOB=ELEMNT/10                                                        0001338
      ROW=ELEMNT-10*JOB                                                    0001339
      GO TO (60,30,70,130,140,150,160,170,180,230,240,270), JOB            0001340
      GO TO 350                                                            0001341
   60 NAME(1)=CHAR                                                         0001342
      NCHAR=1                                                              0001343
      POP=.FALSE.                                                          0001344
      NUM=.FALSE.                                                          0001345
      GO TO 30                                                             0001346
   70 NCHAR=NCHAR+1                                                        0001347
      NAME(NCHAR)=CHAR                                                     0001348
      GO TO 30                                                             0001349
   80 IF (POP) GO TO 100                                                   0001350
      POP=.TRUE.                                                           0001351
      IF (NUM) GO TO 90                                                    0001352
      SWITCH=3                                                             0001353
      GO TO 300                                                            0001354
   90 SWITCH=4                                                             0001355
      GO TO 320                                                            0001356
  100 IF (TOP.EQ.0) GO TO 120                                              0001357
      IF (CODE.LT.STACK(TOP)) GO TO 120                                    0001358
      SWITCH=5                                                             0001359
      VALUE=STACK(TOP)                                                     0001360
      INDEX=VALUE                                                          0001361
      GO TO 330                                                            0001362
  110 TOP=TOP-1                                                            0001363
      GO TO 100                                                            0001364
  120 TOP=TOP+1                                                            0001365
      STACK(TOP)=CODE                                                      0001366
      GO TO 30                                                             0001367
  130 CODE=DIFF                                                            0001368
      GO TO 80                                                             0001369
  140 CODE=PLUS                                                            0001370
      GO TO 80                                                             0001371
  150 CODE=DIV                                                             0001372
      GO TO 80                                                             0001373
  160 CODE=PROD                                                            0001374
```

```
      GO TO 80                                                          0001375
  170 CODE=UNARY                                                        0001376
      GO TO 120                                                         0001377
  180 IF (POP) GO TO 200                                                0001378
      POP=.TRUE.                                                        0001379
      IF (NUM) GO TO 190                                                0001380
      SWITCH=6                                                          0001381
      GO TO 300                                                         0001382
  190 SWITCH=7                                                          0001383
      GO TO 320                                                         0001384
  200 IF (TOP.EQ.0) GO TO 350                                           0001385
      IF (PAREN.EQ.STACK(TOP)) GO TO 220                                0001386
      SWITCH=8                                                          0001387
      VALUE=STACK(TOP)                                                  0001388
      INDEX=VALUE                                                       0001389
      GO TO 330                                                         0001390
  210 TOP=TOP-1                                                         0001391
      GO TO 200                                                         0001392
  220 TOP=TOP-1                                                         0001393
      GO TO 30                                                          0001394
  230 TOP=TOP+1                                                         0001395
      STACK(TOP)=PAREN                                                  0001396
      GO TO 30                                                          0001397
  240 NAME(1)=CHAR                                                      0001398
      NCHAR=1                                                           0001399
      NUM=.TRUE.                                                        0001400
      POP=.FALSE.                                                       0001401
      GO TO 30                                                          0001402
  250 IF (TOP.EQ.0) GO TO 370                                           0001403
      SWITCH=9                                                          0001404
      VALUE=STACK(TOP)                                                  0001405
      IF (VALUE.EQ.PAREN) GO TO 350                                     0001406
      INDEX=VALUE                                                       0001407
      GO TO 330                                                         0001408
  260 TOP=TOP-1                                                         0001409
      GO TO 250                                                         0001410
  270 IVAL=IBLNK                                                        0001411
      IF (NCHAR.EQ.0) GO TO 350                                         0001412
      CALL PACK(NAME,IVAL,NCHAR,4)                                      0001413
      DO 280 J=1,5                                                      0001414
      IF (IVAL.EQ.IFNCTS(J)) GO TO 290                                  0001415
  280 CONTINUE                                                          0001416
      TYPE 410, IVAL                                                    0001417
      GO TO 350                                                         0001418
  290 CODE=J-11                                                         0001419
      TOP=TOP+1                                                         0001420
      STACK(TOP)=CODE                                                   0001421
      GO TO 230                                                         0001422
  300 VARBLE=DBLNK                                                      0001423
      CALL PACK(NAME,VARBLE,NCHAR,8)                                    0001424
      CALL BFIND(&310,VARBLE,INDEX,NAMES,IPNTS,IDIM)                    0001425
      GO TO 330                                                         0001426
  310 TYPE 390, VARBLE                                                  0001427
      GO TO 360                                                         0001428
  320 VALUE=ICONV(NAME,NCHAR,J,ERR)                                     0001429
      IF (ERR) GO TO 350                                                0001430
      INDEX=0                                                           0001431
      IF (J.NE.0) VALUE=VALUE*10.**J                                    0001432
```

```
330 I=I+1                                                              0001433
    IF (I.LE.15) GO TO 340                                             0001434
    TYPE 420                                                           0001435
    GO TO 350                                                          0001436
340 POLISH(I)=VALUE                                                    0001437
    TYPE(I)=INDEX                                                      0001438
    GO TO (250,250,100,100,110,200,200,210,260), SWITCH                0001439
350 TYPE 380, (EXPR(J),J=1,L)                                          0001440
360 ERR=.TRUE.                                                         0001441
370 RETURN                                                             0001442
380 FORMAT (' ERROR IN EXPRESSION: ',50A1/23X,30A1)                    0001443
390 FORMAT (' UNDEFINED NAME ',A8)                                     0001444
410 FORMAT (1X,A5,'IS NOT A PERMISSIBLE FUNCTION. HENCE')              0001445
420 FORMAT (' MORE THAN 15 NAMES AND OPERATORS USED. HENCE')           0001446
    END                                                                0001447
```

SUBROUTINE NAME: PAUSE

Purpose: This routine is used to provide a system-generated pause in output. If a nonblank character is entered by the user, the nonstandard return is taken.

Calling sequence: CALL PAUSE(&n)

Argument:

n—Statement (in caller) to which a branch will be made if a nonblank character is returned by KEYBRD, or if the nonstandard return is taken from KEYBRD.

Subroutine called: KEYBRD

Common data referenced: None

Called by: BDEF, COLPNT, DUMPIT, NAME, ROWPNT

Error checking and reporting: None

Program logic: PAUSE accepts a single (left-justified) character from KEYBRD. If an EOF is sensed or the character is nonblank, take the nonstandard return.

GRASP SOURCE PROGRAM

```
      SUBROUTINE PAUSE(*)                                              0001448
      DATA IBLNK/' '/,IBELL/"034000000000/                             0001449
      TYPE 1,IBELL                                                     0001450
      CALL KEYBRD(&10,I,1)                                             0001451
      IF (I.EQ.IBLNK) RETURN                                           0001452
   10 RETURN 1                                                         0001453
    1 FORMAT(1X,A1)                                                    0001454
      END                                                              0001455
```

FUNCTION NAME: PNTER

Purpose: PNTER is used to look up user-entered character-string or multiple-choice-type values in the value part of "conditions" statements. Lookup is performed in the appropriate dictionary, and the value returned is a pointer to the particular dictionary item. If the value is not found, an error flag is set and zero is returned.

Calling sequence: IPT = PNTER (VALUE, IDIM, NAME, ITYPE,ERR)

Arguments:

VALUE—Unpacked character-string value to be looked up.

IDIM—Length of the string in VALUE.

NAME—Item number for which the character string represents a value.

ITYPE—Item type of item pointed to by NAME.

ERR—Error flag which is set if the value is not found in the dicitionary pointed to by NAME.

Subroutines called: ACCESS, BINTYP, PACK

Common data referenced: None

Called by: RELEXP

Error checking and reporting: If the character-string value is not found, a message is typed, the error flag is set, and zero is returned as the value of PNTER.

Program logic: If the length of the string is given as zero, a value of blank is returned immediately. Otherwise, the string is packed into STRING. The value of ITYPE then determines whether the character-type dictionaries should be accessed (via ACCESS) or the multiple-choice-type dictionaries should be accessed (via BINTYP). If a character-type dictionary is indicated, a call to ACCESS is made, where the fifth parameter has a value of 3. This returns K as the pointer to the first dictionary item. ACCESS is then called, using the value 4 as the fifth parameter (which returns the K'th entry and updates K to point to the next entry), until all entries have been returned or until a match is found. If a match is found, the entry number is returned as a value. Otherwise, zero is returned as a value, and the nonstandard return is taken. If a multiple-choice-type dictionary is indicated, a call to BINTYP returns the possible values in LABEL. The string (equivalenced to BLABEL) is then compared with the items of LABEL.

G R A S P S O U R C E P R O G R A M

```
      INTEGER FUNCTIONPNTER(VALUE,IDIM,NAME,ITYPE,ERR)                  0001456
      DOUBLE PRECISION LABEL(25),BLABEL                                 0001457
      INTEGER VALUE(1),TANK(25),BITEM(15,25),STRING(12)                 0001458
      LOGICAL ERR                                                       0001459
      EQUIVALENCE (STRING(1),BLABEL)                                    0001460
      DATA IBLNK/' '/                                                   0001461
      ERR=.FALSE.                                                       0001462
      N=IBLNK                                                           0001463
      IF (IDIM.EQ.0) GO TO 110                                          0001464
      DO 10 I=1,12                                                      0001465
   10 STRING(I)=IBLNK                                                   0001466
      CALL PACK(VALUE,STRING,IDIM,60)                                   0001467
      DO 20 I=1,12                                                      0001468
      IF (STRING(13-I).NE.IBLNK) GO TO 30                               0001469
   20 CONTINUE                                                          0001470
      GO TO 110                                                         0001471
   30 LENGTH=13-I                                                       0001472
      IF(ITYPE-3) 35,35,80                                              0001473
35      CALL ACCESS(NAME,K,TANK,NUM,3)                                 0001474
      N=0                                                               0001475
40      IF(K.EQ.0) GO TO 60                                            0001476
      N=N+1                                                             0001477
      CALL ACCESS(NAME,K,TANK,NWORDS,4)                                 0001478
      IF (NWORDS.LT.LENGTH) GO TO 40                                    0001479
      DO 50 I=1,LENGTH                                                  0001480
      IF (TANK(I).NE.STRING(I)) GO TO 40                                0001481
   50 CONTINUE                                                          0001482
      GO TO 110                                                         0001483
   60 ERR=.TRUE.                                                        0001484
      TYPE 130, (VALUE(I),I=1,IDIM)                                     0001485
      GO TO 100                                                         0001486
   80 CALL BINTYP(NAME,LABEL,BITEM,L,M)                                 0001487
      DO 90 N=1,M                                                       0001488
      IF (BLABEL.EQ.LABEL(N)) GO TO 110                                 0001489
```

```
 90 CONTINUE                                                                    0001490
    TYPE 140, BLABEL                                                            0001491
    ERR=.TRUE.                                                                  0001492
100 N=0                                                                         0001493
110 PNTER=N                                                                     0001494
    RETURN                                                                      0001495
130 FORMAT (' CHARACTER TYPE VARIABLE DOES NOT ASSUME VALUE:'/1X,60A1)          0001496
140 FORMAT (' BINARY TYPE VARIABLE DOES NOT ASSUME VALUE ',A8)                  0001497
    END                                                                         0001498
```

SUBROUTINE NAME: PREVAL

Purpose: PREVAL acts as an interface between the calling routine (VLIST) and the arithmetic-expression parsing routine PARSE. This interface allows a reduction in the number of dimensions for the variables in /EXPRNS/ which contain the Reverse-Polish form of the arithmetic expressions entered by the user.

Calling sequence: CALL PREVAL(&n,IEXPR,L,KNT)

Arguments:

n—Statement (in calling routine) to which a branch is made if the routine PARSE sets an error flag.

IEXPR—Contains (in unpacked character form) the expression to be parsed.

L—The length of IEXPR.

KNT—Arithmetic expression counter.

Subroutine called: PARSE

Common data referenced: POLISH, ITYPE, LPS in /EXPRNS/

Called by: VLIST

Error checking and reporting: Error flag returned from PARSE is tested.

Program logic:

1. The expression counter KNT is incremented.
2. Call PARSE, passing the input arguments IEXPR, L, and the KNT'th columns of ITYPE, POLISH along with the KNT'th element of LPS, and an error flag.
3. Take the nonstandard return if the error flag ERR has been set.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE PREVAL(*,IEXPR,L,KNT)                                  0001499
      COMMON /EXPRNS/ POLISH,ITYPE,LPS                                  0001500
      DIMENSION POLISH(15,8), ITYPE(15,8), LPS(8), IEXPR(1)             0001501
      LOGICAL ERR                                                       0001502
      KNT=KNT+1                                                         0001503
      CALL PARSE(IEXPR,L,ITYPE(1,KNT),POLISH(1,KNT),LPS(KNT),ERR)       0001504
      IF (ERR) RETURN 1                                                 0001505
      RETURN                                                            0001506
      END                                                               0001507
```

SUBROUTINE NAME: QUIT

Purpose: QUIT performs "wrap-up" processing prior to exiting from the GRASP system. This involves a typed statement regarding the disposition of files created during the current session.

Calling sequence: CALL QUIT(OFILES,NFILES)

Arguments:

OFILES—List of output files created during this session.

NFILES—The number of items in OFILES.

Subroutines called: KEYBRD, RLIST

Common data referenced: None

Called by: DRIVER

Error checking and reporting: The user's response to prompts is checked for validity.

Program logic:

1. The list of created file names is typed, and the user is asked if he would like to save any of them.
2. If so, he is asked to enter a list of numbers corresponding to those files he wishes to save.
3. The system then instructs him how to delete the files he does not wish to save. This routine is provided primarily for bookkeeping. The file-maintenance functions can be performed at the program level on those systems having this capability.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE QUIT(OFILES,NFILES)                                          0001508
      DIMENSION IMAGE(30),LIST(20), KLIST(20)                                 0001509
      INTEGER OFILES(20),FILE,YES,REPLY                                       0001510
      DATA YES,NO/'Y','N'/                                                    0001511
      IF (NFILES.EQ.0) GO TO 100                                              0001512
      TYPE 110, (I,OFILES(I),I=1,NFILES)                                      0001513
      TYPE 120                                                                0001514
   10 TYPE 130                                                                0001515
      CALL KEYBRD(&100,REPLY,1)                                               0001516
      IF (REPLY.EQ.YES) GO TO 40                                              0001517
      IF (REPLY.EQ.NO) GO TO 20                                               0001518
      TYPE 150                                                                0001519
      GO TO 10                                                                0001520
   20 NKILL=NFILES                                                            0001521
      DO 30 I=1,NFILES                                                        0001522
   30 KLIST(I)=I                                                              0001523
      GO TO 80                                                                0001524
   40 IF (NFILES.EQ.1) GO TO 100                                              0001525
      TYPE 160                                                                0001526
   50 CALL KEYBRD(&100,IMAGE,30)                                              0001527
      CALL RLIST(&50,IMAGE,LIST,NSAVE,20)                                     0001528
      IF (NSAVE.EQ.0) GO TO 20                                                0001529
      NKILL=0                                                                 0001530
      DO 70 I=1,NFILES                                                        0001531
      DO 60 J=1,NSAVE                                                         0001532
      IF (LIST(J).EQ.I) GO TO 70                                              0001533
   60 CONTINUE                                                                0001534
      NKILL=NKILL+1                                                           0001535
      KLIST(NKILL)=I                                                          0001536
   70 CONTINUE                                                                0001537
      IF (NKILL) 100,100,80                                                   0001538
8C      TYPE 90,(OFILES(KLIST(I)),I=1,NKILL)                                  0001539
  100 RETURN                                                                  0001540
9C    FORMAT(' ISSUE .DEL COMMANDS FOR THE FOLLOWING FILES:'/(1X,10A6))       0001541
  110 FORMAT (//' THE FOLLOWING FILES HAVE BEEN CREATED ','DURING THIS S0001542
     1ESSION:'/(I5,5X,A6))                                                    0001543
  120 FORMAT (//' DO YOU WISH TO SAVE ANY OF THEM?')                          0001544
  130 FORMAT (' (ENTER YES OR NO): ',$)                                       0001545
  150 FORMAT (' YOUR REPLY WAS NOT UNDERSTOOD.')                              0001546
  160 FORMAT (' ENTER A LIST OF NUMBERS CORRESPONDING TO '/' THOSE FILES0001547
     1 YOU WISH TO SAVE (IE. 1-3,5).')                                        0001548
      END                                                                     0001549
```

SUBROUTINE NAME: RELEXP

Purpose: This subroutine is used to decode the "condition" appearing in IMAGE into the components NAMEPT, RCODE, and IVAL. If it is unsuccessful, an error message is typed and an error flag is set. The "condition" is in unpacked character form and is assumed to be a *name* followed by a *relation* followed by a *value. Name* must be an item name in the current data base (as established by the file command). *Relation* must be one of the following: EQ, equal; LT, less than; GT, greater than; LE, less than or equal; GE, greater than or equal; NE, not equal; BE, between. *Value* must be a number, number pair, character string, set of qualifiers, permissible multiple-choice acronym, or blank. The following table gives valid constructions for "conditions":

Item type	Relation	Value
Integer or real	EQ, LT, GT, LE, GE, NE	Numeric.
	BE	Numeric pair.
Character	EQ, LT, GT, LE, GE, NE	Any printable string.
	BE	Printable string containing comma.
Multiple choice	EQ, NE	Multiple-choice acronym.
Qualified real	EQ, LT, GT, LE, GE, NE	Numeric.
	BE	Numeric pair.
	EQ, NE	Qualifier set[1] in parentheses.

[1] Qualifier set is one or more of the following characters, each of which occur, at most, once; G, H, L, N, T, or blank.

Calling sequence: CALL RELEXP(&n,IMAGE,NAMEPT, RCODE,IVAL,ERR)

Arguments:

n—Statement number (in calling routine) to which a branch will be made if an all-blank condition is detected.

IMAGE—Contains "condition" in unpacked-character form.

NAMEPT—Returned pointer to item name.

RCODE—Returned encoding of relation having the following possible values:

1–7 corresponding to the relations EQ, LT, GT, LE, GE, NE, BE.

11 or 16 corresponding to the relations EQ or NE, applied to a set of qualifiers.

IVAL—Returned as one of the following:

1. Integer or real value.
2. Pointer to a particular entry in the character dictionary associated with the item pointed to by NAMEPT.
3. Bit encoding, giving the position of a particular multiple-choice acronym in the file containing possible acronym values for the item pointed to by NAMEPT.
4. Pointer to the number pair in the common block BTWN which will be used by this instance of the BE relation.
5. Bit encoding of a qualifier set.

ERR—Returned error flag that is set if an error is detected.

Subroutines called: SCAN, BFIND, ICONV, PNTER, PACK

Common data referenced:

NAMES, ITYPE, PNTERS, IDIM in blank common
IVALS, NBE in /BTWN/

Called by: CONDTN

Error checking and reporting:

1. All testing is performed to insure conformity to the table of valid constructs appearing in the preceding "purpose" section.
2. An error flag that may be set by the routines ICONV or PNTER is tested.
3. A nonstandard return from BFIND indicates an invalid name.

   An error message is printed reporting any of the following errors:

   a. Unable to find relation (that is, EQ, LT, GT, LE, GE, NE, BE).
   b. Incorrect qualifier set.
   c. Qualifier codes are referenced in forms other than EQ or NE.
   d. Invalid name as first syntactic unit of condition.
   e. No comma separating a value pair used with the BE relation.

Program logic:

1. A call to SCAN is made to bracket the name as the first syntactic element. If the image is all blank, the nonstandard return is taken.
2. The name is packed into NAME via ENCODE, and BFIND is used to do the lookup. If the name is not found, a message is typed, and the error flag is set.
3. The next call to SCAN brackets the relation. It is packed into REL and tested against the list of valid relations. Note that RCODE is used as the index. If invalid, a message is typed, and the error flag is set.
4. The value part of the condition is then bracketed via the next call to SCAN. If the value field is blank, IVAL is set to blank.
5. Otherwise, the type of name is determined using ITYPE in blank common.
6. The logical variable BE (indicating the "between" relation) is determined. If set, the second value is determined and stored in the BTWN common area, and IVAL is set to point to the BTWN location. The second value determination is logically similar to the first which is described in step 8.
7. If BE was not set, the value element is tested as a qualifier set. If it is one, the appropriate tests are made, and IVAL is bit encoded to show which codes are present. RCODE is, also, incremented by 10 as a flag indicating comparison of qualifier cores.
8. If the value element was not a qualifier set, and the relation was not BE, IVAL is set via a call to ICONV, if type was numeric. Note that for real values, VAL (equivalenced to IVALL) is set. IVAL is then set by IVALL, which shares storage with VAL. For character and multiple-choice types, IVAL is set using the external function PNTER.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE RELEXP(*,IMAGE,NAMEPT,RCODE,IVAL,ERR)                  0001550
      COMMON NAMES,ITYPE,PNTERS,IDIM                                    0001551
      COMMON /BTWN/ IVALS,NBE                                           0001552
      DIMENSION RELS(7), IVALS(2,10), IMAGE(80), ITYPE(500), NAMES(500) 0001553
```

```
      DOUBLE PRECISION NAME,NAMES,DBLNK                                  0001554
      INTEGER FCNAME,FCOP,FCCON,RCODE,PNTER,BLANK,PNTERS(500),E,REL,RELS0001555
     1,COMMA,RPAREN,IQUAL(6)                                             0001556
      LOGICAL ERR,BE                                                     0001557
      EQUIVALENCE (VAL,IVALL),(IQUAL,BLANK)                              0001558
      DATA COMMA,RPAREN,LPAREN,DBLNK,IQUAL/',',')','(','      ',         0001559
     1 ' ','G','H','L','N','T'/,                                         0001560
     1 RELS/'EQ','LT','GT','LE','GE','NE','BE'/                          0001561
      ERR=.FALSE.                                                        0001562
      I=1                                                                0001563
      CALL SCAN(&295,IMAGE,I,FCNAME,LCNAME,1)                            0001564
      NAME=DBLNK                                                         0001565
      CALL PACK(IMAGE(FCNAME),NAME,LCNAME-FCNAME+1,8)                    0001566
      CALL BFIND(&110,NAME,NAMEPT,NAMES,PNTERS,IDIM)                     0001567
      I=LCNAME+1                                                         0001568
      CALL SCAN(&20,IMAGE,I,FCOP,J,1)                                    0001569
      IF(J.NE.FCOP+1) GO TO 20                                           0001570
      REL=BLANK                                                          0001571
      CALL PACK(IMAGE(FCOP),REL,2,4)                                     0001572
      DO 10 RCODE=1,7                                                    0001573
      IF(RELS(RCODE).EQ.REL) GO TO 30                                    0001574
   10 CONTINUE                                                           0001575
   20 TYPE 320                                                           0001576
      GO TO 144                                                          0001577
   30 I=J+1                                                              0001578
      CALL SCAN(&40,IMAGE,I,FCCON,LCCON,2)                               0001579
      GO TO 140                                                          000158C
   40 IVAL=BLANK                                                         0001581
      GO TO 210                                                          0001582
  295 RETURN 1                                                           0001583
  140 J=ITYPE(NAMEPT)                                                    0001584
      BE=RCODE.EQ.7                                                      0001585
      IF(BE) GO TO 220                                                   0001586
      IF(IMAGE(FCCON).NE.LPAREN) GO TO 150                               0001587
      IF(J.NE.5) GO TO 150                                               0001588
      IF(IMAGE(LCCON).EQ.RPAREN) GO TO 142                               0001589
 1405 TYPE 141                                                           0001590
      GO TO 144                                                          0001591
  142 IF(RCODE.EQ.1) GO TO 145                                           0001592
      IF(RCODE.EQ.6) GO TO 145                                           0001593
      TYPE 143                                                           0001594
  144 ERR=.TRUE.                                                         0001595
      GO TO 210                                                          0001596
  145 RCODE=RCODE+10                                                     0001597
      IVAL=0                                                             0001598
      FCCON=FCCON+1                                                      0001599
      LCCON=LCCON-1                                                      0001600
      DO 147 K=FCCON,LCCON                                               0001601
      J=IMAGE(K)                                                         0001602
      DO 146 I=1,6                                                       0001603
      IF(IQUAL(I).EQ.J) GO TO 147                                        0001604
  146 CONTINUE                                                           0001605
      GO TO 1405                                                         0001606
  147 IVAL=IVAL+2**(I-1)                                                 0001607
      GO TO 210                                                          0001608
  110 TYPE 310, NAME                                                     0001609
      GO TO 144                                                          0001610
  150 GO TO (160,170,190,190,170),J                                      0001611
  160 IVAL=ICONV(IMAGE(FCCON),LCCON-FCCON+1,E,ERR)                       0001612
      GO TO 200                                                          0001613
```

```
170 K=ICONV(IMAGE(FCCON),LCCON-FCCON+1,E,ERR)                          0001614
    IF (K.NE.BLANK) GO TO 180                                          0001615
    IVAL=K                                                             0001616
    GO TO 200                                                          0001617
180 VAL=K*10.**E                                                       0001618
    IVAL=IVALL                                                         0001619
    GO TO 200                                                          0001620
190 IVAL=PNTER(IMAGE(FCCON),LCCON-FCCON+1,NAMEPT,J,ERR)                0001621
200 IF (ERR) GO TO 210                                                 0001622
    IF (.NOT.BE) GO TO 210                                             0001623
    NBE=NBE+1                                                          0001624
    IVALS(1,NBE)=IVAL                                                  0001625
    IVAL=NBE                                                           0001626
210 RETURN                                                             0001627
220 DO 230 I=FCCON,LCCON                                               0001628
    IF (IMAGE(I).EQ.COMMA) GO TO 240                                   0001629
230 CONTINUE                                                           0001630
    TYPE 330                                                           0001631
    GO TO 144                                                          0001632
240 GO TO (250,260,280,280,260),J                                      0001633
250 IVAL=ICONV(IMAGE(I+1),LCCON-I,E,ERR)                               0001634
    GO TO 290                                                          0001635
260 K=ICONV(IMAGE(I+1),LCCON-I,E,ERR)                                  0001636
    IF (K.NE.BLANK) GO TO 270                                          0001637
    IVAL=K                                                             0001638
    GO TO 290                                                          0001639
270 VAL=K*10.**E                                                       0001640
    IVAL=IVALL                                                         0001641
    GO TO 290                                                          0001642
280 IVAL=PNTER(IMAGE(I+1),LCCON-I,NAMEPT,J,ERR)                        0001643
290 IF (ERR) GO TO 210                                                 0001644
    IVALS(2,NBE+1)=IVAL                                                0001645
    LCCON=I-1                                                          0001646
    GO TO 150                                                          0001647
141 FORMAT(' QUALIFIER CODES MUST BE ONE OR MORE OF "L, ,N,T,G,H"',    0001648
   1 ' AND ENCLOSED IN PARENTHESIS.')                                  0001649
143 FORMAT(' ONLY EQ/NE CAN BE USED WITH QUALIFIER CODES.')            0001650
310 FORMAT (' INVALID NAME = ',A8)                                     0001651
320 FORMAT(' UNABLE TO FIND RELATION (LT,GT,LE,GE,EQ,NE,BE)')          0001652
330 FORMAT (' NO COMMA SEPARATING CONSTANTS FOLLOWING BE OPERATOR')    0001653
    END                                                                0001654
```

SUBROUTINE NAME: RETRVE

Purpose: RETRVE is used to retrieve records from a selected file and write them on some other selected file. User-specified encoded retrieval criteria are passed to RETRVE via its argument list.

Calling sequence: CALL RETRVE(&n,&m,IFILES,OFILES, NFILES,POLISH,LPS,VARS,CODES,VALS,NCOND)

Arguments:

n—Statement number (in calling routine) to which a branch is made if previously undetected (by LOGEXP) errors are encountered.

m—Statement number (in calling routine) to which a branch is made if KEYBRD senses an EOF.

IFILES, OFILES—Arrays of input and output file names, respectively.

NFILES—Number of elements in either IFILES or OFILES (IFILES and OFILES are of equal size).

POLISH—Array containing the Reverse-Polish form of the logic expression to be used for data retrieval.

LPS—The number of elements in POLISH.

VARS, CODES, VALS—Arrays that give an encoding of the conditional expressions entered by the user.

NCOND—Number of elements in VARS, CODES, and VALS.

Subroutines called: KEYBRD, OBEY, GETPUT, COMP, UNCODE, PACK

Common data referenced:

FNAMES, SELECT in /FILNAM/
ITYPE in blank common
INPUT, OUTPUT in /IOUNIT/

Called by: DRIVER

Error checking and reporting:

1. Check to assure entry of retrieval criteria (that is, LPS>0)
2. Check to assure absence of undetected errors in the retrieval criteria.

Messages are typed corresponding to the two error situations above.

Program logic:

1. LPS is checked to insure that retrieval criteria have been entered.
2. The elements of VALS are moved to IVAL to allow the equivalencing necessary for mixed modes (in particular, integer and real) possible in retrieval criteria.
3. Input- and output-file names are prompted and accepted, then associated with FORTRAN unit numbers via calls to KEYBRD and OBEY. The new file names are added to IFILES and OFILES.
4. The input file is then read, one record at a time (via GETPUT), until the nonstandard (EOF) return is taken. After each call to GETPUT, the record (IREC) is tested against the retrieval criteria indicated in POLISH, VARS, CODES, and VALS via the logical valued push-down stack technique described as follows: Any element of POLISH less than 27 points to one of the conditional expressions encoded in VARS, CODES, VALS. That expression is evaluated via a reference to the logical function COMP, and the result is placed in the push-down stack. If POLISH(I)>26, it points to one of the logical operators "and," "or," "not" (denoted by *, +, —). If the operator is —, the "not" operation is performed on the topmost stack element. If the operator is * or +, the operation is performed on the topmost two stack elements, the size of the stack is decremented, and the resultant (logical) value replaces the new topmost stack element. After the last element of POLISH has been processed, the size of the stack should be 1, and the value of this element indicates whether or not the record meets the retrieval criteria. If so, it is added to the output file by a call to GETPUT. Counts of records read and records retrieved are kept and typed at the end of retrieval processing.

GRASP SOURCE PROGRAM

```
      SUBROUTINE RETRVE(*,*,IFILES,OFILES,NFILES,POLISH,LPS,VARS,CODES,  0001655
     1VALS,NCOND)                                                        0001656
      COMMON NAMES,ITYPE,IPTS,IDIM                                       0001657
      COMMON /IOUNIT/ INPUT,OUTPUT                                       0001658
      COMMON /FILNAM/ FNAMES,SELECT,PAD                                  0001659
      DIMENSION VAL(26)                                                  0001660
      DOUBLE PRECISION NAMES(500)                                        0001661
      INTEGER IVAL(26),CODES(1),VARS(1),WHICH,DREC(500),OUTPUT,TOP,POL   0001662
     1ISH(1),ITYPE(500),ONE,TWO,VALS(1),EQUATE(4),IPTS(500),SELECT       0001663
      INTEGER FILEID,DBLANK,IFILES(1),OFILES(1),DFAULT,FNAMES(21)        0001664
      LOGICAL COMP,VALUE,STACK(20),EVAL,PAD(4)                           0001665
      EQUIVALENCE (IREC,REC),(IVAL,VAL),(STACK,EVAL),(EQUATE(4),FILEID)  0001666
      DATA ONE,TWO,DBLANK/'1','2',' ',  EQUATE/'EQUA','TE 1',2*' '       0001667
      IF (LPS.GT.0) GO TO 10                                             0001668
      TYPE 180                                                           0001669
      GO TO 170                                                          0001670
   10 DO 20 I=1,NCOND                                                    0001671
   20 IVAL(I)=VALS(I)                                                    0001672
      DFAULT=FNAMES(SELECT)                                              0001673
      NRECS=0                                                            0001674
      NFOUND=0                                                           0001675
```

```
    NFILES=NFILES+1                                                    0001676
    EQUATE(3)=ONE                                                      0001677
    TYPE 190                                                           0001678
    CALL KEYBRD(&175,DREC,5)                                           0001679
    CALL PACK(DREC,FILEID,5,5)                                         0001680
    IF (FILEID.EQ.DBLANK) FILEID=DFAULT                                0001681
    IFILES(NFILES)=FILEID                                              0001682
    CALL OBEY(&170,EQUATE,4)                                           0001683
    EQUATE(3)=TWO                                                      0001684
    TYPE 210                                                           0001685
    CALL KEYBRD(&175,DREC,5)                                           0001686
    CALL PACK(DREC,FILEID,5,5)                                         0001687
    OFILES(NFILES)=FILEID                                              0001688
    CALL OBEY(&170,EQUATE,4)                                           0001689
 30 CALL GETPUT(&140,DREC,1)                                           0001690
    NRECS=NRECS+1                                                      0001691
    TOP=0                                                              0001692
    DO 110 J=1,LPS                                                     0001693
    INDEX=POLISH(J)                                                    0001694
    IF (INDEX.GT.26) GO TO 70                                          0001695
    TOP=TOP+1                                                          0001696
    WHICH=VARS(INDEX)                                                  0001697
    I=ITYPE(WHICH)                                                     0001698
    IREC=DREC(WHICH)                                                   0001699
    GO TO (40,50,40,60,62), I                                          0001700
 40 STACK(TOP)= COMP(IREC       ,IVAL(INDEX),D1,D2,CODES(INDEX),1)     0001701
    GO TO 110                                                          0001702
 50 STACK(TOP)= COMP(ID1,ID2,REC        ,VAL(INDEX),CODES(INDEX),2)     0001703
    GO TO 110                                                          0001704
 60 STACK(TOP)= COMP(IREC       ,IVAL(INDEX),D1,D2,CODES(INDEX),3)     0001705
    GO TO 110                                                          0001706
 62 IF(IREC.NE.DBLANK) GO TO 63                                        0001707
    IF(CODES(INDEX).LT.11) GO TO 50                                    0001708
    STACK(TOP)=.FALSE.                                                 0001709
    GO TO 110                                                          0001710
 63 REC=UNCODE(REC,IQ)                                                 0001711
    IF(CODES(INDEX)-11) 50,64,66                                       0001712
 64 STACK(TOP)=MOD(IVAL(INDEX)/2**(IQ-1),2).EQ.1                       0001713
    GO TO 110                                                          0001714
 66 STACK(TOP)=MOD(IVAL(INDEX)/2**(IQ-1),2).EQ.0                       0001715
    GO TO 110                                                          0001716
 70 VALUE=STACK(TOP)                                                   0001717
    IF (INDEX-30) 80,90,100                                            0001718
 80 TOP=TOP-1                                                          0001719
    STACK(TOP)=STACK(TOP).OR.VALUE                                     0001720
    GO TO 110                                                          0001721
 90 TOP=TOP-1                                                          0001722
    STACK(TOP)=STACK(TOP).AND.VALUE                                    0001723
    GO TO 110                                                          0001724
100 STACK(TOP)=.NOT.STACK(TOP)                                         0001725
110 CONTINUE                                                           0001726
    IF (TOP.EQ.1) GO TO 120                                            0001727
    TYPE 220                                                           0001728
    REWIND INPUT                                                       0001729
    RETURN 1                                                           0001730
120 IF (EVAL) GO TO 130                                                0001731
    GO TO 30                                                           0001732
130 CALL GETPUT(&140,DREC,2)                                           0001733
    NFOUND=NFOUND+1                                                    0001734
    GO TO 30                                                           0001735
```

```
140 TYPE 230, NRECS,IFILES(NFILES)                                          0001736
    IF (NFOUND.GT.0) GO TO 150                                              0001737
    TYPE 240                                                                0001738
    NFILES=NFILES-1                                                         0001739
    GO TO 160                                                               0001740
150 TYPE 250, NFOUND,FILEID                                                 0001741
160 REWIND INPUT                                                            0001742
    REWIND OUTPUT                                                           0001743
170 RETURN                                                                  0001744
175 RETURN 2                                                                0001745
180 FORMAT (' LOGIC MUST BE SUPPLIED BEFORE A RETRIEVAL CAN BE MADE')       0001746
190 FORMAT (' ENTER INPUT FILE NAME: ',$)                                   0001747
210 FORMAT (' ENTER OUTPUT FILE NAME: ',$)                                  0001748
220 FORMAT (' ERROR IN LOGIC EXPRESSION')                                   0001749
230 FORMAT (' ALL',I6,' RECORDS OF',A6,' SEARCHED.')                        0001750
240 FORMAT (' THERE ARE NO RECORDS WHICH SATISFY THE REQUEST')              0001751
250 FORMAT (I10,' RECORDS FOUND WHICH SATISFY THE REQUEST.'/' THEY HAV0001752
   1E BEEN STORED IN ',A6)                                                  0001753
    END                                                                     0001754
```

SUBROUTINE NAME: RLIST

Purpose: RLIST is used to convert an unpacked character string representing a list of user-entered numbers into a corresponding numeric list.

Calling sequence: CALL RLIST(&n,IMAGE,LIST,NUMC, MOST)

Arguments:

n—Statement number (in caller) to which a branch will be made if an uncorrectable error in the entered character string is detected.

IMAGE—The unpacked character-string form of the list of numbers.

LIST—The list of numbers which are returned in numeric form.

NUMC—The number of items in LIST.

MOST—The maximum number of items that LIST may contain.

Subroutines called: None

Common data referenced: None

Called by: DEFLST, QUIT

Error checking and reporting: If an illegal character (not 0–9, dash, or comma) is detected, an error message is typed and the nonstandard return is taken. If the list isn't composed of numbers or number ranges separated by commas, an error message is typed, and the nonstandard return is taken.

Program logic: Each character of the unpacked string is processed via the following transition matrix:

	0—9	,	—
1.	f(1)/2	error	error
2.	f(2)/2	f(3)/1	f(3)/3
3.	f(1)/4	error	error
4.	f(2)/4	f(4)/1	error

f(i) are defined as:

f(1)—Start a number value.

f(2)—Build number value by adding digit on right.

f(3)—Number built, add it to list.

f(4)—Fill LIST with values up to and including current value.

Rows 1 and 2 are used to process a single list element or the first of a number-range pair.

Rows 3 and 4 are used to process the second of a number-range pair. Blanks are completely ignored.

The columns of the transition matrix are associated with the indicated characters, and the rows correspond to individual states.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE RLIST(*,IMAGE,LIST,NUMC,MOST)                          0001755
      INTEGER LIST(1),IMAGE(30),CHAR(12),COL(12),IMAT(4,3)              0001756
      DATA IMAT/12,22,14,24,0,31,0,41,0,33,0,0/,IBLNK/' '/              0001757
      DATA CHAR/'0','1','2','3','4','5','6','7','8','9',',','-'/       0001758
      DATA COL/10*1,2,3/                                                0001759
      NUMC=0                                                            0001760
      LAST=1                                                            0001761
      IP=0                                                              0001762
      IROW=1                                                            0001763
   10 IP=IP+1                                                           0001764
      IF (IP.GT.30) GO TO 120                                           0001765
      ICHAR=IMAGE(IP)                                                   0001766
      IF (ICHAR.EQ.IBLNK) GO TO 10                                      0001767
      DO 20 I=1,12                                                      0001768
      IF (ICHAR.EQ.CHAR(I)) GO TO 40                                    0001769
   20 CONTINUE                                                          0001770
      TYPE 150, ICHAR                                                   0001771
   30 RETURN 1                                                          0001772
   35 TYPE 145,IVAL                                                     0001773
      GO TO 30                                                          0001774
   40 IFS=IMAT(IROW,COL(I))                                             0001775
      IF (IFS.NE.0) GO TO 60                                            0001776
   50 TYPE 160                                                          0001777
      GO TO 30                                                          0001778
   60 IROW=MOD(IFS,10)                                                  0001779
      IFS=IFS/10                                                        0001780
      GO TO (70,80,90,100), IFS                                         0001781
   70 IVAL=I-1                                                          0001782
      GO TO 10                                                          0001783
   80 IVAL=10*IVAL+I-1                                                  0001784
      GO TO 10                                                          0001785
```

```
 90 NUMC=NUMC+1                                                          0001786
    IF (NUMC.GT.MOST) GO TO 110                                          0001787
    IF(IVAL.GT.MOST) GO TO 35                                            0001788
    LIST(NUMC)=IVAL                                                      0001789
    GO TO (10,140), LAST                                                 0001790
100 LIST(NUMC+1)=LIST(NUMC)+1                                            0001791
    NUMC=NUMC+1                                                          0001792
    IF (NUMC.GT.MOST) GO TO 110                                          0001793
    IF(IVAL.GT.MOST) GO TO 35                                            0001794
    IF (LIST(NUMC).LT.IVAL) GO TO 100                                    0001795
    GO TO (10,140), LAST                                                 0001796
110 TYPE 170, MOST                                                       0001797
    NUMC=MOST                                                            0001798
    GO TO 140                                                            0001799
120 IF (NUMC.NE.0) GO TO 130                                             0001800
    IF (IROW.EQ.1) GO TO 140                                             0001801
    NUMC=1                                                               0001802
    LIST(1)=IVAL                                                         0001803
    GO TO 140                                                            0001804
130 IF (MOD(IROW,2).NE.0) GO TO 50                                       0001805
    LAST=2                                                               0001806
    IROW=IROW/2                                                          0001807
    GO TO (90,100), IROW                                                 0001808
140 RETURN                                                               0001809
145 FORMAT(I5,' DOES NOT CORRESPOND TO A CATEGORY. RE-ENTER NUMBERS')    0001810
150 FORMAT (1X,A1,' IS AN ILLEGAL CHARACTER. RE-ENTER NUMBERS.')         0001811
160 FORMAT (' EACH NUMBER OR NUMBER RANGE (IE. 4-7) EXCEPT ','THE LAST0001812
   1 MUST BE FOLLOWED BY A COMMA.'/' RE-ENTER NUMBERS.')                 0001813
170 FORMAT (' TOO MANY NUMBERS GIVEN. ONLY THE FIRST',I3,' WILL BE USE0001814
   1D.')                                                                 0001815
    END                                                                  0001816
```

SUBROUTINE NAME: ROWPNT

Purpose: ROWPNT prints selected items or expressions, one to a line, from a selected file.

Calling sequence: CALL ROWPNT(&n,NPAGE)

Arguments:

n—Statement (in caller) to which VLIST will branch when an EOF is encountered by KEYBRD.

NPAGE—Number of lines per page of printed output.

Subroutines called: VLIST, ACCESS, GETPUT, PAUSE, EVAL, UNCODE, BINTYP, BLIST

Common data referenced:

POLISH, ICODE, LPS in /EXPRNS/

ITYPE in blank common

Called by: LIST

Error checking and reporting: None

Program logic:

1. The list of items to be printed is obtained via a call to VLIST.
2. A call to ACCESS is made to initialize the lookup of dictionary-type items.
3. Each record of the selected file is obtained via GETPUT and a line counter is incremented and tested. If it exceeds the page size, a pause is generated via PAUSE.
4. After each record is obtained, the selected items are evaluated (if necessary) and printed.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE ROWPNT(*,NPAGE)                                          0001817
      COMMON NAMES,ITYPE,IPTS,IPAD                                        0001818
      COMMON /EXPRNS/ POLISH,ICODE,LPS                                    0001819
      DOUBLE PRECISION NAMES,LABEL,NAMESI,VNAMES(20)                      0001820
      INTEGER BLANK,TANK,IPTS(500),IQUAL(6),POLISH(15,8),                 0001821
     1 ICODE(15,8),LPS(8)                                                 0001822
      DIMENSION ITYPE(500), BITEM(15,25), ITEMS(20), IREC(500),           0001823
     1      REC(500), NAMES(500), TANK(25), LABEL(25), LIST(25)           0001824
      LOGICAL ERR                                                         0001825
      EQUIVALENCE (REC,IREC),(IVAL,VAL),(LIST,TANK),(BLANK,IQUAL)         0001826
      DATA IQUAL/' ','G','H','L','N','T'/                                 0001827
      KOUNT=0                                                             0001828
      CALL VLIST(&270,VNAMES,ITEMS,NUM)                                   0001829
      IF (NUM.EQ.0) GO TO 260                                             0001830
      TYPE 280                                                            0001831
      CALL ACCESS(II,IVAL,TANK,J,1)                                       0001832
  120 CALL GETPUT(&260,IREC,1)                                            0001833
      KOUNT=KOUNT+NUM+1                                                   0001834
      IF (KOUNT.LE.NPAGE) GO TO 130                                       0001835
      KOUNT=0                                                             0001836
      CALL PAUSE(&260)                                                    0001837
  130 DO 240 JJ=1,NUM                                                     0001838
      II=ITEMS(JJ)                                                        0001839
      IF(II.GT.0) GO TO 135                                               0001840
      II=-II                                                              0001841
      NAMESI=VNAMES(JJ)                                                   0001842
      VAL=EVAL(IREC,ICODE(1,II),POLISH(1,II),LPS(II),ERR)                 0001843
      IF(ERR) GO TO 240                                                   0001844
      GO TO 150                                                           0001845
  135 IVAL=IREC(II)                                                       0001846
      IF (IVAL.EQ.BLANK) GO TO 240                                        0001847
      KIND=ITYPE(II)                                                      0001848
      NAMESI=VNAMES(JJ)                                                   0001849
      GO TO (140,150,160,220,170), KIND                                   0001850
  140 TYPE 300, NAMESI,IVAL                                               0001851
      GO TO 240                                                           0001852
  150 TYPE 310, NAMESI,VAL                                                0001853
      GO TO 240                                                           0001854
  160 CALL ACCESS(II,IVAL,TANK,J,2)                                       0001855
      TYPE 320, NAMESI,(TANK(I),I=1,J)                                    0001856
      GO TO 240                                                           0001857
```

```
170 VAL=UNCODE(VAL,IQ)                                              0001858
    TYPE 310,NAMESI,VAL,IQUAL(IQ)                                   0001859
    GO TO 240                                                       0001860
220 CALL BINTYP(II,LABEL,BITEM,K,M)                                 0001861
    KOUNT=KOUNT+1                                                   0001862
    TYPE 340, NAMESI                                                0001863
    CALL BLIST(LIST,NUMS,IVAL)                                      0001864
    DO 230 I=1,NUMS                                                 0001865
    KOUNT=KOUNT+1                                                   0001866
    J=LIST(I)                                                       0001867
230 TYPE 330, LABEL(J),(BITEM(L,J),L=1,K)                           0001868
240 CONTINUE                                                        0001869
    TYPE 290                                                        0001870
    GO TO 120                                                       0001871
260 RETURN                                                          0001872
270 RETURN 1                                                        0001873
280 FORMAT (///)                                                    0001874
290 FORMAT (1X,3(8H********))                                       0001875
300 FORMAT (1X,A8,1H=,I9)                                           0001876
310 FORMAT (1X,A8,1H=,1PG12.5,A1)                                   0001877
320 FORMAT (1X,A8,1H=,12A5/15X,12A5)                                0001878
330 FORMAT (5X,A8,15A4)                                             0001879
340 FORMAT (1X,A8,1H=)                                              0001880
    END                                                             0001881
```

SUBROUTINE NAME: SCAN

Purpose: This subroutine is used to set character-position pointers for the syntactic elements of a condition (name, relation, value(s)) or a logical expression.

Calling sequence: CALL SCAN(&n,IMAGE,IS,I1,I2,IT)

Arguments:

n—Statement number (in calling routine) to which a branch will be made if IMAGE is all blanks.
IMAGE—String of unpacked left-justified characters.
IS—Starting position of the scan.
I1—Pointer to first character of syntactic element.
I2—Pointer to last character of syntactic element.
IT—Embedded blank switch.

Subroutines called: None
Common data referenced: None
Called by: RELEXP, LOGEXP
Error checking and reporting: None
Program logic:

1. The position of the first nonblank character is determined.
2. If all characters after the IS'th are blank, the nonstandard return is taken.
3. If no embedded blanks are permitted (IT=1), the position of the last nonblank character to the right of the position found in step 1 is determined, and control passes to the caller.
4. Otherwise (if, IT=2), the position of the first nonblank character to the left of position 80 is determined and control returns to the caller.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE SCAN(*,IMAGE,IS,I1,I2,IT)                          0001882
      INTEGER IMAGE(1)                                              0001883
      DATA IBLNK/' '/                                               0001884
      DO 1 I=IS,80                                                  0001885
      IF(IMAGE(I).NE.IBLNK) GO TO 2                                 0001886
    1 CONTINUE                                                      0001887
      RETURN 1                                                      0001888
    2 I1=I                                                          0001889
      J=I1+1                                                        0001890
      GO TO (3,10),IT                                               0001891
    3 DO 4 I2=J,80                                                  0001892
      IF(IMAGE(I2).EQ.IBLNK) GO TO 5                                0001893
    4 CONTINUE                                                      0001894
      I2=81                                                         0001895
    5 I2=I2-1                                                       0001896
    6 RETURN                                                        0001897
   10 DO 11 I=J,80                                                  0001898
      I2=80-I+J                                                     0001899
      IF(IMAGE(I2).NE.IBLNK) GO TO 6                                0001900
   11 CONTINUE                                                      0001901
      GO TO 6                                                       0001902
      END                                                           0001903
```

SUBROUTINE NAME: START

Purpose: START determines availability of data bases and their associated files.

Calling sequence: CALL START

Arguments: None

Subroutine called: IFILE

Common data referenced:
All variables in /FILNAM/ except NUMF

Called by: DRIVER

Error checking and reporting: None

Program logic:

1. The name GFILE is associated with FORTRAN input unit 20.
2. A welcoming message is typed and records of GFILE are read to fill the /FILNAM/ common area.
3. As each record is read, parts of it are output to the terminal.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE START                                                  0001904
      COMMON /FILNAM/ MASTER,MASK,DEFTN,DFILE,BFILE,NUMF,NUMI,IDIMS     0001905
      DOUBLE PRECISION CONTNT(4)                                        0001906
      INTEGER MASTER(4),MASK(4),DEFTN(4),DFILE(4),BFILE(4),IDIMS(4)     0001907
      CALL IFILE(20,'GFILE')                                            0001908
      NUMF=1                                                            0001909
      TYPE 1                                                            0001910
   10 READ(20,11,END=20) MASTER(NUMF),CONTNT,MASK(NUMF),                0001911
     1 DEFTN(NUMF),DFILE(NUMF),BFILE(NUMF),IDIMS(NUMF)                  0001912
      TYPE 12,MASTER(NUMF),CONTNT                                       0001913
      NUMF=NUMF+1                                                       0001914
      GO TO 10                                                          0001915
   20 NUMF=NUMF-1                                                       0001916
      REWIND 20                                                         0001917
      TYPE 2                                                            0001918
      RETURN                                                            0001919
    1 FORMAT(/' WELCOME TO THE USGS GRASP RETRIEVAL SYSTEM.'/           0001920
     1 ' AT THE CURRENT TIME THE FOLLOWING DATA BASES ARE AVAILABLE:')   0001921
    2 FORMAT(/' BEFORE ANY OF THESE DATA BASES MAY BE ACCESSED,'/       0001922
     1 ' THE "FILE" COMMAND SHOULD BE USED TO IDENTIFY THE DATA',       0001923
     2 ' BASE OF INTEREST.')                                            0001924
   11 FORMAT(A5,1X,4A10,4(1X,A5),I4)                                    0001925
   12 FORMAT(/1X,A6,'- ',4A10)                                          0001926
      END                                                               0001927
```

FUNCTION NAME: UNCODE

Purpose: UNCODE breaks down each qualified real value into a real value and a qualifier code.

Calling sequence: VALUE=UNCODE(VAL,ID)

Arguments:

VAL—Packed value and qualifier.

ID—Encoding of qualifier value.

Subroutines called: None

Common data referenced: None

Called by: COLPNT, DUMPIT, FIT, MEAN, RETRVE, ROWPNT, EVAL

Error checking and reporting: None

Program logic: The type REAL argument VAL may be visualized as composed of both whole and fractional parts. ID is set to the 3 low-order bits of the whole part. The whole part is then shifted right 3 bits, and the result is added to the fractional part to form the value returned by the function.

GRASP SOURCE PROGRAM

```
      FUNCTION UNCODE(VAL,ID)                                    0001928
      RSIGN=SIGN(1.0,VAL)                                        0001929
      VAL=ABS(VAL)                                               0001930
      IPART=VAL                                                  0001931
      REST=VAL-IPART                                             0001932
      ID=MOD(IPART,8)                                            0001933
      UNCODE=RSIGN*((IPART/8)+REST)                              0001934
      RETURN                                                     0001935
      END                                                        0001936
```

SUBROUTINE NAME: VLIST

Purpose: VLIST prompts for and accepts (via KEYBRD) a set of item names or arithmetic expressions that will be printed by the caller. The user is provided the ability to reuse the list that was last entered.

Calling sequence: CALL VLIST(&m,VNAMES,LIST,N)

Arguments:

m—Statement number (in caller) to which a branch will be made if an EOF is sensed by KEYBRD.
VNAMES—List of item names to be printed.
LIST—Item numbers corresponding to VNAMES.
N—Number of elements in VNAMES and LIST.

Subroutines called: KEYBRD, SCAN, BFIND, PREVAL, PACK

Common data referenced:

NAMES, PNTS, IDIM in blank common

Called by: BDEF, COLPNT, ROWPNT

Error checking and reporting: None

Program logic:

1. The user is asked if he wishes to enter a new list of item names. If not, the values of the arguments as set by a previous call are returned. Otherwise, a new list is processed as below.
2. The routine requests the user to enter a set of item names or expressions. Each entry is processed as specified below until a blank entry is detected. Control is then returned to the caller.
3. An entry is accepted via KEYBRD, and leading and/or trailing blanks are eliminated via SCAN.
4. If the entry length is greater than seven characters, an expression is assumed. If the entry is not found to be an item name (via BFIND), an expression is assumed.
5. If the entry is determined to be an item name, that name is stored in VNAMES, and the corresponding item number is stored in LIST.
6. If the entry is an expression, a call to PREVAL is made to parse it into Reverse-Polish form for later evaluation. KNT points to the Reverse-Polish form (stored in /EXPNS/ by PREVAL), and the negative of KNT is returned in LIST.
7. The values of the arguments are saved for future calls if required.

G R A S P S O U R C E P R O G R A M

```
      SUBROUTINE VLIST(*,VNAMES,LIST,N)                                 0001937
      COMMON NAMES,TYPE,PNTS,IDIM                                       0001938
      DOUBLE PRECISION NAMES(500),NAME,BLANK,VNAMES(1),VSAVE(20)        0001939
      INTEGER PNTS(500),TYPE(500),LIST(1),EXPR(80),HALVES(2),EXPHDG(20) 0001940
     1 ,LSAVE(20)                                                       0001941
      EQUIVALENCE (HALVES(1),NAME)                                      0001942
      DATA IEQUAL,BLANK/'=','      '/,NSAVE,NO/0,'N'/                   0001943
      DATA EXPHDG/'1.','2.','3.','4.','5.','6.','7.','8.','9.','10.','110001944
     1.','12.','13.','14.','15.','16.','17.','18.','19.','20.'/        0001945
      IF(NSAVE.EQ.0) GO TO 5                                            0001946
      TYPE 4                                                            0001947
      CALL KEYBRD(&110,I,1)                                             0001948
      IF(I.NE.NO) GO TO 5                                               0001949
      N=NSAVE                                                           0001950
      DO 6 I=1,NSAVE                                                    0001951
      LIST(I)=LSAVE(I)                                                  0001952
    6 VNAMES(I)=VSAVE(I)                                                0001953
      GO TO 130                                                         0001954
    5 N=0                                                               0001955
      KNT=0                                                             0001956
      TYPE 120                                                          0001957
   10 N=N+1                                                             0001958
      TYPE 140, EXPHDG(N)                                               0001959
      CALL KEYBRD(&110,EXPR,80)                                         0001960
      CALL SCAN(&115,EXPR,1,L1,L,2)                                     0001961
      IF(L-L1.GT.6) GO TO 60                                            0001962
      L2=L1+6                                                           0001963
      NAME=BLANK                                                        0001964
      CALL PACK(EXPR(L1),NAME,L2-L1+1,8)                                0001965
      CALL BFIND(&60,NAME,I,NAMES,PNTS,IDIM)                            0001966
   30 VNAMES(N)=NAME                                                    0001967
      LIST(N)=I                                                         0001968
      IF (N.EQ.20) RETURN                                               0001969
      GO TO 10                                                          0001970
```

```
 60 IF (KNT.EQ.8) RETURN                                                0001971
    NAME=BLANK                                                          0001972
    DO 70 K=1,10                                                        0001973
    IF (EXPR(K+L1-1).EQ.IEQUAL) GO TO 80                                0001974
 70 CONTINUE                                                            0001975
    K=1                                                                 0001976
    HALVES(2)=EXPHDG(N)                                                 0001977
    GO TO 90                                                            0001978
 80 J=MIN0(K-1,7)+L1-1                                                  0001979
    CALL PACK(EXPR(L1),NAME,J -L1+1,8)                                  0001980
    K=K+1                                                               0001981
 90 CALL PREVAL(&100,EXPR(K+L1-1),L,KNT)                                0001982
    I=-KNT                                                              0001983
    GO TO 30                                                            0001984
100 N=N-1                                                               0001985
    GO TO 10                                                            0001986
110 RETURN 1                                                            0001987
115 N=N-1                                                               0001988
    NSAVE=N                                                             0001989
    DO 116 I=1,NSAVE                                                    0001990
    LSAVE(I)=LIST(I)                                                    0001991
116 VSAVE(I)=VNAMES(I)                                                  0001992
130 RETURN                                                              0001993
  4 FORMAT(' DO YOU WISH TO ENTER A NEW LIST OF NAMES OR',              0001994
   1 ' EXPRESSIONS? (YES OR NO): ',$)                                   0001995
120 FORMAT (' ENTER THE NAMES OF ITEMS OR THE EXPRESSIONS ',            0001996
   1'WHICH YOU WANT PRINTED.')                                          0001997
140 FORMAT (1X,A4,$)                                                    0001998
    END                                                                 0001999
```